Advanced Reporting Guide:
Enhancing Your Business Intelligence Application with MicroStrategy Desktop
Tenth Edition, July 2005, version 8.0.1

To ensure that you are using the documentation that corresponds to the software you are licensed to use, compare this version number with the software version shown in “About MicroStrategy...” in the Help menu of your software.

Document number: 09450801

Copyright © 2005 by MicroStrategy Incorporated. All rights reserved.

If you have not executed a written or electronic agreement with MicroStrategy or any authorized MicroStrategy distributor, the following terms apply:

This software and documentation are the proprietary and confidential information of MicroStrategy Incorporated and may not be provided to any other person. Copyright © 2001-2005 by MicroStrategy Incorporated. All rights reserved.

This SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" AND WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND BY EITHER MICROSTRATEGY INCORPORATED OR ANYONE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DISTRIBUTION OF THE SOFTWARE OR DOCUMENTATION, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, GOOD TITLE AND NONINFRINGEMENT, QUALITY OR ACCURACY. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE AND DOCUMENTATION IS WITH YOU. SHOULD THE SOFTWARE OR DOCUMENTATION PROVE DEFECTIVE, YOU (AND NOT MICROSTRATEGY, INC. OR ANYONE ELSE WHO HAS BEEN INVOLVED WITH THE CREATION, PRODUCTION, OR DISTRIBUTION OF THE SOFTWARE OR DOCUMENTATION) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

In no event will MicroStrategy, Inc. or any other person involved with the creation, production, or distribution of the Software be liable to you on account of any claim for damage, including any lost profits, lost savings, or other special, incidental, consequential, or exemplary damages, including but not limited to any damages assessed against or paid by you to any third party, arising from the use, inability to use, quality, or performance of such Software and Documentation, even if MicroStrategy, Inc. or any such other person or entity has been advised of the possibility of such damages, or for the claim by any other party. In addition, MicroStrategy, Inc. or any other person involved in the creation, production, or distribution of the Software shall not be liable for any claim by you or any other party for damages arising from the use, inability to use, quality, or performance of such Software and Documentation, based upon principles of contract warranty, negligence, strict liability for the negligence of indemnity or contribution, the failure of any remedy to achieve its essential purpose, or otherwise. The entire liability of MicroStrategy, Inc. and your exclusive remedy shall not exceed, at the option of MicroStrategy, Inc., either a full refund of the price paid, or replacement of the Software. No oral or written information given out expands the liability of MicroStrategy, Inc. beyond that specified in the above limitation of liability. Some states do not allow the limitation or exclusion of liability for incidental or consequential damages, so the above limitation may not apply to you.

The information contained in this manual (the Documentation) and the Software are copyrighted and all rights are reserved by MicroStrategy, Inc. MicroStrategy, Inc. reserves the right to make periodic modifications to the Software or the Documentation without obligation to notify any person or entity of such revision. Copying, duplicating, selling, or otherwise distributing any part of the Software or Documentation without prior written consent of an authorized representative of MicroStrategy, Inc. are prohibited. U.S. Government Restricted Rights. It is acknowledged that the Software and Documentation were developed at private expense, that no part is public domain, and that the Software and Documentation are Commercial Computer Software provided with RESTRICTED RIGHTS under Federal Acquisition Regulations and agency supplements to them. Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFAR 252.227-7013 et. seq. or subparagraphs (c)(1) and (2) of the Commercial Computer Software—Restricted Rights at FAR 52.227-19, as applicable. Contractor is MicroStrategy, Inc. 1861 International Drive, McLean, Virginia 22102. Rights are reserved under copyright laws of the United States with respect to unpublished portions of the Software.

The following are either trademarks or registered trademarks of MicroStrategy Incorporated in the United States and certain other countries:

Various MicroStrategy products contain the copyrighted technology of third parties. This product may contain one or more of the following copyrighted technologies:
Graph Generation Engine Copyright © 1998-2004. Three D Graphics, Inc. All rights reserved.
Actuate® Formula One. Copyright © 1993-2004 Actuate Corporation. All rights reserved.
XML parser Copyright © 2003 Microsoft Corporation. All rights reserved.
Xalan XSLT processor. Copyright © 1999-2004. The Apache Software Foundation. All rights reserved.
Xerces XML parser. Copyright © 1999-2004. The Apache Software Foundation. All rights reserved.
FOP XSL formatting objects. Copyright © 2004. The Apache Software Foundation. All rights reserved.
Portions of Intelligence Server memory management Copyright 1991-2004 Compuware Corporation. All rights reserved.
International Components for Unicode
Copyright © 1999-2004 Compaq Computer Corporation
Copyright © 1999-2004 Hewlett-Packard Company
Copyright © 1999-2004 IBM Corporation
Copyright © 1999-2004 Hummingbird Communications Ltd.
Copyright © 1999-2004 Silicon Graphics, Inc.
Copyright © 1999-2004 Sun Microsystems, Inc.
Copyright © 1999-2004 The Open Group
All rights reserved.
Real Player and RealJukebox are included under license from Real Networks, Inc. Copyright © 1999-2004. All rights reserved.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document description</td>
<td>xix</td>
</tr>
<tr>
<td>Who should use this guide</td>
<td>xx</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>xx</td>
</tr>
<tr>
<td>Objectives</td>
<td>xx</td>
</tr>
<tr>
<td>About this book</td>
<td>xxi</td>
</tr>
<tr>
<td>Typographical standards</td>
<td>xxii</td>
</tr>
<tr>
<td>For online and printed documentation</td>
<td>xxii</td>
</tr>
<tr>
<td>For printed documentation only</td>
<td>xxiii</td>
</tr>
<tr>
<td>Resources</td>
<td>xxv</td>
</tr>
<tr>
<td>Product documentation</td>
<td>xxv</td>
</tr>
<tr>
<td>Installed documentation</td>
<td>xxv</td>
</tr>
<tr>
<td>International support</td>
<td>xxviii</td>
</tr>
<tr>
<td>User assistance</td>
<td>xxviii</td>
</tr>
<tr>
<td>Online help</td>
<td>xxix</td>
</tr>
<tr>
<td>Technical Support</td>
<td>xxix</td>
</tr>
<tr>
<td>Feedback</td>
<td>xxxiv</td>
</tr>
</tbody>
</table>

### 1. Introduction to Advanced Reporting

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Basic MicroStrategy terminology</td>
<td>2</td>
</tr>
<tr>
<td>Source systems</td>
<td>2</td>
</tr>
<tr>
<td>ETL process</td>
<td>2</td>
</tr>
<tr>
<td>Data warehouse</td>
<td>3</td>
</tr>
<tr>
<td>Logical data model</td>
<td>4</td>
</tr>
<tr>
<td>Physical warehouse schema</td>
<td>4</td>
</tr>
<tr>
<td>Metadata database</td>
<td>5</td>
</tr>
<tr>
<td>Facts</td>
<td>6</td>
</tr>
</tbody>
</table>
2. Reports

Introduction ................................................................................ 13
Before you begin................................................................. 14
   Reporting Essentials review ........................................ 14
The basic report ................................................................. 17
Filtering .................................................................................. 19
   What is a filter? ................................................................. 19
   What is a report limit? ..................................................... 20
   The difference between report filters and report limits .... 22
   What is a metric qualification? ................................. 26
   What is report as filter? ................................................. 30
Understanding how a report is executed ................................ 31
   Data definition and view definition objects ................. 32
   Intelligent Cubes ............................................................. 33
View filters ........................................................................... 35
   What is a view filter? .................................................... 35
Derived metrics ...................................................................... 38
   What is a derived metric? ........................................... 38
Dynamic aggregation ........................................................... 40
   What is dynamic aggregation? .............................. 40
View filter effects ................................................................ 42
   The effect of view filters on metrics ......................... 42
   The effect of view filters on dynamic aggregation ...... 43
   Metric qualification in the view filter ...................... 46
View definition in the report execution cycle ...................... 49
Exceptions to dynamic aggregation .................................... 50
Subtotals ............................................................................. 54
   What are subtotals? ..................................................... 54
   What are custom subtotals? ....................................... 60
   What are user-defined subtotals? ........................... 63
   What are smart subtotals? ........................................ 71
Shortcut metrics ................................................................ 73
   What are shortcut metrics? .................................... 73
Advanced Reporting Guide

Contents

Advanced sorting ......................................................................... 74
Formatting .................................................................................... 78
Formatting Cells dialog box ................................................... 78
Formatting layers ..................................................................... 87
Order of layers ....................................................................... 92
Autostyles .............................................................................. 97
Deployment .................................................................................. 98
Project progression ................................................................ 98
Predesigned reports ............................................................ 102
Deploying predesigned reports ............................................ 105
Shortcut to a filter ................................................................. 108
Shortcut to a template .......................................................... 109
Object templates .................................................................. 114
Evaluation order ......................................................................... 118
Default evaluation order ....................................................... 119
Specified evaluation order ................................................... 120
Evaluation order in data definition and view definition ....... 120
Find and replace ........................................................................ 127
Bulk export ................................................................................. 130

3. Creating Freeform SQL Reports

Introduction .............................................................................. 133
Freeform SQL reporting ............................................................. 136
When should I use the Freeform SQL reporting feature?.... 136
SQL query syntax ................................................................ 137
SQL support ......................................................................... 138
Freeform SQL reports vs. standard reports ......................... 140
Freeform SQL reports in Report Services documents .......... 141
Reporting features .................................................................... 143
Filters ................................................................................... 143
Prompts ............................................................................... 143
Drilling .................................................................................. 147
Security for data access ............................................................ 148
Access control list ................................................................ 148
Security filters ...................................................................... 149
Managed objects ....................................................................... 153
Creating Freeform SQL reports ................................................. 158
Creating a Freeform SQL report from a database .......... 158
Creating a Freeform SQL report from an Excel file .......... 160
Creating a Freeform SQL report from a text file .......... 163
Creating a Freeform SQL report using a stored procedure . 166
4. Creating OLAP Cube Reports

**Introduction** ................................................................. 169

- MicroStrategy integration with SAP BW ........................................ 171
- Understanding MicroStrategy architecture ................................ 172
- Understanding the SAP BW terminology ..................................... 176
- Relating SAP BW objects to MicroStrategy objects ...................... 179
  - SAP BW variables .......................................................... 185
  - SAP BW structures ......................................................... 188
- Using the OLAP Cube Catalog .................................................. 188
  - Importing cubes .............................................................. 189
  - Mapping cubes .................................................................. 191
- Reporting features ..................................................................... 197
  - Filters ............................................................................. 197
  - Prompts ........................................................................... 199
  - Drilling ............................................................................ 201
  - Setting display hierarchy .................................................... 202

**Related features** ..................................................................... 202

- Managed objects ...................................................................... 203
- Authentication ......................................................................... 203

**Connecting to SAP BW servers** ................................................. 204

- Windows ............................................................................... 204
- UNIX .................................................................................... 205

**Creating OLAP cube reports** ..................................................... 208

5. Filters

**Introduction** ............................................................................. 211

- Types of filters ......................................................................... 212
- Report filter options ................................................................. 212
- Attribute qualification ............................................................... 213
  - Attribute element list qualification ...................................... 213
  - Attribute form qualification ................................................. 215
  - Attribute-to-attribute qualification ...................................... 218
  - Attribute Qualification Prompt .......................................... 220
- Set qualification ......................................................................... 220
  - Set qualification: metric qualification ................................... 221
  - Set qualification: relationship qualification .......................... 226
  - Metric qualification prompt ................................................. 227
- Shortcut to a report .................................................................... 227
  - Report Object Prompt ....................................................... 228
- Shortcut to a filter ..................................................................... 228
  - Filter Object Prompt ......................................................... 229
Advanced qualification: custom expression............................... 229
Advanced qualification: relationship filters........................... 229
Advanced qualification: apply functions............................... 230
Advanced qualification: joint element list ............................. 231

6. Metrics

Introduction.............................................................................. 235
Metric types ............................................................................... 236
Simple metrics .................................................................... 236
Nested metrics .................................................................. 238
Compound metrics............................................................... 239
Derived metrics.................................................................... 240
Distinguishing between simple and compound metrics ...... 240
Definition of simple metrics........................................................ 241
Formula................................................................................ 242
Level .................................................................................... 244
Example 1: Using level metrics.................................................. 259
Example 2: Using level metrics.................................................. 265
Example 3: Removing report level ................................................ 271
Condition.............................................................................. 272
Transformation..................................................................... 275
Definition of compound metrics ................................................. 276
Smart metrics....................................................................... 277
Evaluation order................................................................. 279
Metric aggregation and subtotals............................................... 279
Subtotals.............................................................................. 280
Dynamic aggregation........................................................... 281
Join specification ................................................................. 282
Inner joins versus outer joins ............................................... 282
Formula join type for compound metrics.............................. 284
Joins between metrics ......................................................... 285
Metric-specific VLDB properties................................................ 285
Metric VLDB properties........................................................ 287
Analytical Engine VLDB properties for metrics .................... 288
Metric column alias ............................................................. 290
Formatting metrics ..................................................................... 291
Number display codes ......................................................... 292
Symbols and their functions.................................................. 293
Colors .................................................................................. 296
Creating metrics in the Report Editor......................................... 296
Derived metrics ................................................................. 297
Shortcut metrics ................................................................. 299
Useful functions .................................................................. 303
  Rank ............................................................................. 303
  Count .......................................................................... 304
  Running and moving sums and averages ......................... 304
  N-tile ......................................................................... 305
  First and Last ................................................................. 305
Creating metrics in the Command Manager ......................... 305
  Operators and functions ................................................... 306
  Level ........................................................................... 307
  Level filtering .................................................................. 308
  Level grouping .................................................................. 309
  Additional level capabilities ............................................. 311
  Pass-through functions .................................................... 312

7. Data Mining Services

   Introduction ..................................................................... 313
   Data Mining Services ......................................................... 314
     Approaches for data mining with MicroStrategy ......... 315
     The Data Mining Services Workflow ......................... 320
     Predictive metrics and performance ............................. 321
   Creating a dataset report ................................................. 322
     Data mining datasets .................................................... 323
     Guidelines for creating a dataset report ..................... 325
     Predictive input metrics .............................................. 327
     Using non-MicroStrategy datasets ............................... 334
   Creating a predictive model ............................................. 335
     Using third-party applications ..................................... 335
     Using MicroStrategy .................................................. 336
   Importing the predictive model ...................................... 344
     Data mining function parameters ................................. 347
     Returning confidences/probabilities instead of scores . 348
     Aggregating predictive metrics ................................... 349
   Using the predictive metric ............................................. 350
     Using the predictive metric in reports ....................... 350
     Using the predictive metric in other objects .............. 351
     Predictive Model Viewer ............................................ 351
   Example ........................................................................ 352
### 8. Custom Groups and Consolidations

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>359</td>
</tr>
<tr>
<td>Custom groups</td>
<td>360</td>
</tr>
<tr>
<td>Benefits of using a custom group</td>
<td>361</td>
</tr>
<tr>
<td>Banding qualification</td>
<td>362</td>
</tr>
<tr>
<td>Example: banding points</td>
<td>364</td>
</tr>
<tr>
<td>Custom group elements</td>
<td>367</td>
</tr>
<tr>
<td>Custom group element headers</td>
<td>368</td>
</tr>
<tr>
<td>Custom group options</td>
<td>368</td>
</tr>
<tr>
<td>Sorting custom groups</td>
<td>371</td>
</tr>
<tr>
<td>Sorting by metric values of items</td>
<td>372</td>
</tr>
<tr>
<td>Changing the position of totals</td>
<td>373</td>
</tr>
<tr>
<td>Changing the position of element headers</td>
<td>374</td>
</tr>
<tr>
<td>Custom groups and SQL</td>
<td>375</td>
</tr>
<tr>
<td>Example: custom groups</td>
<td>376</td>
</tr>
<tr>
<td>Consolidations</td>
<td>377</td>
</tr>
<tr>
<td>Create a “virtual” attribute</td>
<td>378</td>
</tr>
<tr>
<td>Perform row level math</td>
<td>379</td>
</tr>
<tr>
<td>Consolidation elements</td>
<td>380</td>
</tr>
<tr>
<td>Elements of the same attribute</td>
<td>381</td>
</tr>
<tr>
<td>Elements from different levels</td>
<td>382</td>
</tr>
<tr>
<td>Elements from unrelated attributes</td>
<td>383</td>
</tr>
<tr>
<td>Existing elements</td>
<td>383</td>
</tr>
<tr>
<td>Importing elements</td>
<td>383</td>
</tr>
<tr>
<td>Evaluation order</td>
<td>384</td>
</tr>
<tr>
<td>Consolidations and SQL</td>
<td>385</td>
</tr>
<tr>
<td>Example: consolidations</td>
<td>386</td>
</tr>
<tr>
<td>Custom group and consolidation comparison</td>
<td>387</td>
</tr>
<tr>
<td>Arithmetic operations</td>
<td>388</td>
</tr>
<tr>
<td>Site of final calculation</td>
<td>389</td>
</tr>
<tr>
<td>SQL efficiency</td>
<td>389</td>
</tr>
<tr>
<td>Recursive definition</td>
<td>389</td>
</tr>
<tr>
<td>Display mode</td>
<td>390</td>
</tr>
<tr>
<td>Subtotals</td>
<td>390</td>
</tr>
</tbody>
</table>

### 9. Prompts

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>391</td>
</tr>
<tr>
<td>What is a prompt?</td>
<td>392</td>
</tr>
<tr>
<td>Prompt search ability</td>
<td>393</td>
</tr>
<tr>
<td>Prompt properties</td>
<td>393</td>
</tr>
<tr>
<td>Types of prompts</td>
<td>394</td>
</tr>
</tbody>
</table>
Contents

Filter definition prompts ........................................................ 394
Example: filter definition prompt ........................................... 397
Object prompts .................................................................... 397
Example: object prompt ....................................................... 398
Value prompts ...................................................................... 399
Example: value prompt ........................................................ 400
Level prompts ...................................................................... 400
Example: level prompt ......................................................... 401
Saving reports with prompts ...................................................... 401
Example: basic prompts ....................................................... 402
System prompts ......................................................................... 402

10. Facts

Introduction ................................................................. 405
What is a fact? ........................................................................... 405
Fact structure ............................................................................. 406
Fact definition ...................................................................... 407
Fact expressions .................................................................. 408
Column alias ........................................................................ 410
Level extensions .................................................................. 411
Defining facts ............................................................................. 420
Example: fact definition ........................................................ 420

11. Attributes

Introduction ................................................................. 421
What is an attribute? .......................................................... 422
Attribute elements .............................................................. 423
Attribute forms .................................................................. 425
Attribute form properties .................................................. 426
Attribute form expressions ................................................ 427
Attributes and SQL .............................................................. 430
Column alias ........................................................................ 431
Form groups .............................................................................. 432
Attribute relationships ........................................................ 433
Joint child relationships ....................................................... 433
Attribute display ...................................................................... 434
Using attribute forms versus characteristic attributes .......... 435
Compound attributes ........................................................... 435
Example: creating a compound attribute ............................. 436
# Advanced Reporting Guide

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>12. HTML Documents</strong></td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>437</td>
</tr>
<tr>
<td>HTML document layout</td>
<td>438</td>
</tr>
<tr>
<td>Advanced concepts: XML and XSL</td>
<td>439</td>
</tr>
<tr>
<td>XML</td>
<td>439</td>
</tr>
<tr>
<td>XSL</td>
<td>440</td>
</tr>
<tr>
<td>Creating HTML documents</td>
<td>441</td>
</tr>
<tr>
<td>HTML document views</td>
<td>442</td>
</tr>
<tr>
<td>Report characteristics</td>
<td>442</td>
</tr>
<tr>
<td>Image URLs</td>
<td>443</td>
</tr>
<tr>
<td>Best practices for creating dashboards</td>
<td>445</td>
</tr>
<tr>
<td>Layout</td>
<td>445</td>
</tr>
<tr>
<td>Parameters for dashboard design</td>
<td>445</td>
</tr>
<tr>
<td>Implementing gauge-based dashboards</td>
<td>448</td>
</tr>
<tr>
<td>Example: implementing a gauge-based dashboard</td>
<td>449</td>
</tr>
<tr>
<td>XSL samples for simple customization</td>
<td>452</td>
</tr>
<tr>
<td>Example: building an HTML document</td>
<td>454</td>
</tr>
<tr>
<td><strong>13. Hierarchies</strong></td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>455</td>
</tr>
<tr>
<td>Types of hierarchies</td>
<td>456</td>
</tr>
<tr>
<td>System hierarchy</td>
<td>456</td>
</tr>
<tr>
<td>User hierarchies</td>
<td>457</td>
</tr>
<tr>
<td>Hierarchy tools</td>
<td>457</td>
</tr>
<tr>
<td>Hierarchy organization</td>
<td>458</td>
</tr>
<tr>
<td>Hierarchy structure</td>
<td>459</td>
</tr>
<tr>
<td>Hierarchy display</td>
<td>460</td>
</tr>
<tr>
<td>Locked hierarchy</td>
<td>460</td>
</tr>
<tr>
<td>Limited hierarchy</td>
<td>461</td>
</tr>
<tr>
<td>Filtered hierarchy</td>
<td>462</td>
</tr>
<tr>
<td>Entry point</td>
<td>463</td>
</tr>
<tr>
<td>Hierarchy browsing</td>
<td>464</td>
</tr>
<tr>
<td>Drilling down using hierarchies</td>
<td>466</td>
</tr>
<tr>
<td><strong>14. Drill Maps</strong></td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>467</td>
</tr>
<tr>
<td>What is drilling?</td>
<td>467</td>
</tr>
<tr>
<td>Drill maps and drill paths</td>
<td>468</td>
</tr>
<tr>
<td>Default drill paths</td>
<td>468</td>
</tr>
<tr>
<td>Creating custom drill maps and paths</td>
<td>469</td>
</tr>
<tr>
<td>Drill map association</td>
<td>474</td>
</tr>
</tbody>
</table>
### Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.</td>
<td>Logical Tables</td>
<td>479</td>
</tr>
<tr>
<td>16.</td>
<td>Data Marting</td>
<td>503</td>
</tr>
<tr>
<td>17.</td>
<td>Transformations</td>
<td>511</td>
</tr>
<tr>
<td>18.</td>
<td>Aggregate Tables</td>
<td>519</td>
</tr>
</tbody>
</table>

#### Logical Tables

- **Introduction** .................................................. 479
- Logical tables ................................................. 480
- How should I use logical tables? ......................... 481
- Creating logical tables ....................................... 482
  - Using SQL for logical views ............................... 483
- Logical view examples ....................................... 483
  - Business case 1: Distinct attribute lookup table ... 483
  - Business case 2: Attribute form expression across multiple tables .......................... 485
  - Business case 3: Slowly changing dimensions ....... 486
  - Business case 4: One-to-many transformation tables .......................... 497
  - Business case 5: Outer joins between attribute lookup tables... 498

#### Data Marting

- **Introduction** .................................................. 503
- Associated terminology ...................................... 503
- Sample business scenarios ................................... 504
  - The output of data mart reports: relational tables ... 505
  - Custom groups in data mart tables ....................... 507

#### Transformations

- **Introduction** .................................................. 511
- What is a transformation? ..................................... 512
- Transformation metrics ....................................... 513
  - Transformation metrics and joint child attributes .... 514
- Transformation components ................................ 515
- Example: transformations ..................................... 517

#### Aggregate Tables

- **Introduction** .................................................. 519
- Why should I use aggregate tables? ....................... 520
- Aggregation terminology .................................... 521
  - Aggregation versus pre-aggregation ................... 521
  - Degree of aggregation .................................... 524
- When should I use aggregate tables? ..................... 524
  - Frequency of queries at the level ....................... 525

---

Levels of drill map association ............................................. 475
Removing associations ................................................... 477
Advanced Reporting Guide

Contents

Relationship between the parent and child ...................... 526
Compression ratio ....................................................... 527
Integrating aggregate tables ........................................ 527
Logical table size .................................................... 528

19. Partition Mapping

Introduction ................................................................. 529
Server versus application partitioning .............................. 530
    Server-level partitioning ......................................... 530
    Application-level partitioning ................................ 530
Metadata partition mapping ......................................... 531
    Homogenous and heterogeneous partitions ................ 531
    Data slices .......................................................... 532
    Attribute qualifications ........................................... 533
Warehouse partition mapping ........................................ 533
Metadata versus warehouse partition mapping ............... 534

A. MicroStrategy Tutorial

Introduction ................................................................. 535
What is the MicroStrategy Tutorial? .............................. 535
The MicroStrategy Tutorial data model ......................... 538
    Geography hierarchy ............................................. 540
    Products hierarchy ............................................... 542
    Customers hierarchy ............................................ 544
    Time hierarchy .................................................... 546
    Promotions hierarchy .......................................... 547
The MicroStrategy Tutorial schema ............................... 549
    Geography schema ............................................... 551
    Products schema .................................................. 552
    Customers schema ................................................ 553
    Time schema ....................................................... 554
    Promotions schema .............................................. 554
    Sales fact tables ................................................... 555
    Inventory fact tables ............................................. 556
    Miscellaneous fact tables ..................................... 556

B. Data types

Description ............................................................... 559
Mapping data sources to MicroStrategy data types ......... 560
Format types ............................................................ 561
Big Decimal .............................................................. 562
C. Pass-through Expressions

- **Description** ................................................................. 565
  - The Apply functions .................................................. 566
  - Function syntax .......................................................... 567
  - Argument types .......................................................... 568
  - Upgrading database types ......................................... 568
  - Changing database types ........................................ 568
- **Syntax examples** .......................................................... 569
  - ApplySimple .............................................................. 569
  - ApplyAgg .................................................................... 570
  - ApplyOLAP ................................................................. 571
  - ApplyComparison ....................................................... 571
  - ApplyLogic ................................................................. 572

D. Advanced Data Modeling

- **Introduction** ................................................................. 573
  - Attribute relationship ................................................ 574
  - Many-to-many relationships ....................................... 575
    - Loss of analytical capability .................................. 575
    - Multiple counting ................................................. 577
    - Working with many-to-many relationships ............... 580
  - Joint child relationships .......................................... 583
    - What is a joint child relationship? ......................... 583
    - Supporting joint child relationships ..................... 584
  - Attribute roles .......................................................... 586
    - Automatic attribute role recognition ................... 588
    - Explicit table aliasing ........................................... 589

E. Logical and Mathematical Operators for Filtering

- **Introduction** ................................................................. 591
  - What is an operator? .................................................. 592
    - Logical operators .................................................. 592
    - Comparison operators .......................................... 595
    - Rank and percent operators ................................... 597
    - Pattern operators .................................................. 598

F. Warehouse Catalog SQL

- **Introduction** ................................................................. 599
  - Customizing Catalog SQL statements ....................... 600
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>The table name space .................. 600</td>
</tr>
<tr>
<td>SQL template strings and incomplete Catalog SQL ............... 601</td>
</tr>
<tr>
<td>Structure of Catalog Table SQL .................................................................. 601</td>
</tr>
<tr>
<td>Structure of Full Catalog SQL ................................................................. 602</td>
</tr>
<tr>
<td>Default Warehouse Catalog SQL ................................................................. 603</td>
</tr>
</tbody>
</table>

**G. Project Creation Assistant**

**Introduction** ............................................................................................... 607

Before you begin ............................................................................................... 608

Plan your project ............................................................................................... 608

Create the metadata database ........................................................................... 609

Project creation ................................................................................................. 609

Initialize/create the project ............................................................................. 610

Select tables from the Warehouse Catalog ...................................................... 610

Create facts ........................................................................................................ 613

Create attributes .................................................................................................. 614

Project completion .............................................................................................. 616

Additional schema configurations ...................................................................... 616

**H. ETL Information**

**Description** ................................................................................................ 619

Report ETL information ....................................................................................... 620

**I. Desktop Commands**

**Introduction** ................................................................................................. 621

Background .......................................................................................................... 622

Why would you use Desktop Commands? ......................................................... 622

Setting the Desktop homepage .......................................................................... 623

Viewing the Desktop commands ....................................................................... 625

Commands ............................................................................................................ 626

ChangeView ........................................................................................................... 626

Editor ...................................................................................................................... 628

Execute .................................................................................................................. 628

ExecuteDocument .................................................................................................. 629

ExecuteReport ....................................................................................................... 630

Open ....................................................................................................................... 631

Reset ....................................................................................................................... 633

Shortcut ................................................................................................................. 633
Document description

The MicroStrategy Advanced Reporting Guide provides comprehensive information on advanced topics in the MicroStrategy query and reporting products. This guide assumes and continues to build on a basic understanding of information provided in the Basic Reporting Guide.

It uses Business Scenarios to provide examples of each concept illustrated. The MicroStrategy Tutorial, MicroStrategy’s sample warehouse, metadata, and project are at the center of each of these examples. Information about the MicroStrategy Tutorial may be found in Introduction to MicroStrategy.

By the end of this document, you will have an understanding of the important concepts required to build sophisticated reports using the MicroStrategy platform.
Who should use this guide

This document is designed for

- Report Designers who will be creating advanced reports and reporting objects such as templates, metrics, filters, drill maps, and so on
- Project Designers who will be creating advanced schema objects such as facts, attributes, hierarchies, and so on
- Analysts who will be performing advanced report manipulations

Prerequisites

Before working with this document, you should be familiar with

- projects, attributes, facts, and metrics
- simple project and report creation
- SQL statements

Objectives

After reading this manual, you will be able to

- understand the difference between the view definition and the data definition of a report, and know which report execution steps belong to each
- create advanced metrics using functionality such as conditionality, dimension level, and transformation
- understand what Catalog SQL is and how to use it
- create facts using column aliases, level extensions, fact relations, and other advanced fact concepts
- create advanced attributes
About this book

This book is divided into chapters and reference appendices. The chapters provide concepts about individual topics, such as metrics, data marting, hierarchies, and so on.

Each chapter begins with a brief overview of the content. The chapter is then divided into subsections organized in the best method to promote learning. If applicable, a series of steps is provided to carry out the task description and facilitate the learning process.

When you need specific information about a task, use the table of contents or index to locate the information quickly.
Typographical standards

For online and printed documentation

MicroStrategy online and hard copy documentation follows presentation conventions and cues to help you locate, identify, and understand important concepts and procedures. The following table lists these conventions.

<table>
<thead>
<tr>
<th>Type</th>
<th>Indicates</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bold</strong></td>
<td>• button names, check boxes, dialog boxes, options, lists, and menus that are the focus of actions or part of a list of such GUI elements and their definitions</td>
</tr>
<tr>
<td></td>
<td>• text to be entered by the user</td>
</tr>
<tr>
<td><strong>italic</strong></td>
<td>• new terms defined within the text and in the glossary</td>
</tr>
<tr>
<td></td>
<td>• names of other product manuals</td>
</tr>
<tr>
<td></td>
<td>• when part of a command syntax, indicates variable information to be replaced by the user</td>
</tr>
<tr>
<td><strong>Courier font</strong></td>
<td>• calculations</td>
</tr>
<tr>
<td></td>
<td>• code samples</td>
</tr>
<tr>
<td></td>
<td>• registry keys</td>
</tr>
<tr>
<td></td>
<td>• path and file names</td>
</tr>
<tr>
<td></td>
<td>• URLs</td>
</tr>
<tr>
<td></td>
<td>• messages displayed in the screen</td>
</tr>
<tr>
<td><strong>UPPERCASE</strong></td>
<td>• keyboard command key (such as ENTER)</td>
</tr>
<tr>
<td></td>
<td>• shortcut key (such as CTRL+V)</td>
</tr>
<tr>
<td><strong>+</strong></td>
<td>A keyboard command that calls for the use of more than one key (for example, SHIFT+F1)</td>
</tr>
</tbody>
</table>
For printed documentation only

The following are explanations of the font style changes, icons, and different types of notes that you see in this guide.

**Actions**

References to screen elements and keys that are the focus of actions are in bold Arial font style. Following is an example:

1. Click **Select Warehouse**.

**Code**

References to code, formulas, or calculations within paragraphs are formatted in regular Courier New font style. Following is an example:

\[ \text{Sum(revenue)/number of months.} \]

**Data entry**

References to literal data you must type in an exercise or procedure are in bold Arial font style. References to data you type in that could vary from user to user or system to system are in bold italic Arial font style. Following is an example:

Type `cmdmgr -f scriptfile.scp` and press ENTER.

Type `copy c:\filename d:\foldername\filename`

**Keyboard keys**

References to a keyboard key or shortcut keys are in uppercase letters. Following is an example:

To bold the selected text, press CTRL+B.
New terms

New terms to note are in regular italic font style. These terms are defined when they are first encountered in the course material. Following is an example:

The *aggregation level* is the level of calculation for the metric.

Notes and warnings

A note icon indicates helpful information.

A warning icon calls your attention to very important information that should be read before continuing the course.

Heading icons

The following heading icons are used to indicate specific practice and review sections:

Precedes a Case Study. Case Studies are real-life examples of companies that are using MicroStrategy.

Resources

Product documentation

MicroStrategy documentation includes a full set of product manuals designed to help you find the information you need to install, configure, design, and administer your Business Intelligence and Narrowcast Server, as well as full SDK documentation to help you extend MicroStrategy and integrate it with your existing applications.

A list of documentation links is available to access all documentation installed from your CD-ROM. Most of these documents have been provided in Acrobat Portable Document format (PDF).

Adobe Acrobat Reader 4.0 is required to view these documents. If you do not have Acrobat Reader installed on your computer, you can download it from www.adobe.com.

Installed documentation

To access an installed manual

1. From the Windows Start menu, choose Programs, MicroStrategy, then Product Manuals. A Web page opens with a list of available manuals in PDF format.

2. Click the link for the desired manual.

3. Some information is provided in HTML help format. When you select one of these guides, the File Download dialog box opens. Select Open this file from its current location, and click OK.
If bookmarks are not visible on the left side of an Acrobat document, click the **Bookmarks and Page** from the **View** menu, then select the topic and section you want to see. You can also scroll from the title page of the guide to its table of contents, and select from there the topic you want to read.

The following documents are provided on your CD-ROM in Acrobat Portable Document format (PDF):

**MicroStrategy Overview**

- *MicroStrategy Quick Start Guide*

**Manuals for Query, Reporting, and Analysis Products**

- *MicroStrategy Installation and Configuration Guide*
- *MicroStrategy Upgrade Guide*
- *MicroStrategy Basic Reporting Guide*
- *MicroStrategy Advanced Reporting Guide*
- *MicroStrategy Document Creation Guide*
- *MicroStrategy System Administration Guide*
- *MicroStrategy Analytical Functions Reference*
- *MicroStrategy Web SDK*

The Web SDK is available in the MicroStrategy Developer Library, which is sold as part of the MicroStrategy SDK.

**Manuals for Information Delivery and Alerting Products**

- *MicroStrategy Narrowcast Server Getting Started Guide*
• MicroStrategy Narrowcast Server Installation and Configuration Guide

• MicroStrategy Narrowcast Server Application Designer Guide

• MicroStrategy Narrowcast Server System Administrator Guide

• MicroStrategy Narrowcast Server Upgrade Guide

Manuals for Analytics Modules

• Business Intelligence Developer Kit (BIDK) Installation and Porting Guide

• Customer Analysis Module Reference

• Sales Force Analysis Module Reference

• Web Traffic Analysis Module Reference

• Financial Reporting Analysis Module Reference

• Sales and Distribution Analysis Module Reference

• Human Resources Analysis Module Reference

Software Development Kits

• MicroStrategy Developer Library

• MicroStrategy Web SDK

  The Web SDK is available in the MicroStrategy Developer Library, which is sold as part of the MicroStrategy SDK.

• Narrowcast Server SDK Guide
International support

MicroStrategy supports several locales. Support for a locale typically includes native database and operating system support, support for date formats, decimal formats, currency symbols, etc. and availability of translated interfaces and documentation. The level of support is defined in terms of the components of a MicroStrategy Business Intelligence environment. A MicroStrategy Business Intelligence environment consists of the following components, collectively known as a “configuration”:

- warehouse, metadata, and statistics databases
- MicroStrategy Intelligence Server
- MicroStrategy Web Server
- MicroStrategy Desktop client
- Web browser

MicroStrategy is certified in homogeneous configurations (where all the components lie in the same locale) in the following languages—English (US), French, German, Italian, Japanese, Korean, Portuguese (Brazilian), Spanish, Chinese (simplified) and Swedish.

MicroStrategy also provides limited support for heterogeneous configurations (where some of the components may lie in different locales). Please contact MicroStrategy Technical Support for more details.

A translated user interface is available in each of the above languages. In addition, translated versions of the online help files and product documentation are available in several of the above languages.

User assistance

The following paragraphs describe the types of assistance available to answer questions you may have regarding MicroStrategy products.
Online help

MicroStrategy provides several modes of access to online help:

- From the Help menu, by selecting Contents and Index to see the main table of contents for the help system
- By pressing F1 to see context-sensitive help addressing the function or task you are currently performing

Technical Support

If you have questions about a specific MicroStrategy product, you should:

1. Consult the product guides, online help, readme files, and release notes

2. Consult the online knowledge base at http://www.microstrategy.com/support/k_base/index.asp

   A technical administrator in your organization can probably help you resolve some of your issues immediately.

3. If the resources listed in steps 1 and 2 do not provide you with a solution, contact MicroStrategy Technical Support directly. To ensure the most effective and productive relationship with MicroStrategy Technical Support, review the Policies and Procedures document posted at http://www.microstrategy.com/Support/Policies. Please refer to the terms of your purchase agreement to determine the type of support available to you.

   The table on the following page shows where, when, and how to contact MicroStrategy Technical Support. If you are unable to reach MicroStrategy Technical Support by phone during the hours of operation, you have the option to leave a voicemail message or send electronic mail.
<table>
<thead>
<tr>
<th>Region</th>
<th>Contact Information</th>
</tr>
</thead>
</table>
| **North America**                  | E-mail: support@microstrategy.com  
Web: https://support.microstrategy.com  
Fax: (703) 848–8710  
Phone: (703) 848–8700  
Message: (703) 848-8709  
Hours: 9:00 A.M.–7:00 P.M. Eastern Time (1400–0000 GMT), Monday–Friday except holidays |
| **Europe, the Middle East, and Africa (EMEA)** | E-mail: eurosupp@microstrategy.com  
Web: https://support.microstrategy.com  
Fax: +44 (0) 208 396 0001  
The European Technical Support Centre is closed on certain public holidays. These holidays reflect the national public holidays in each country.  
Phone:  
- United Kingdom: +44 (0) 208 396 0085  
- Benelux: +31 20 346 9210  
- Finland: +35 8 9 6937 9620  
- France: +33 1 41 91 86 49  
- Germany: +49 69 95096206  
- Ireland: +35 3 1242 1522  
- Italy: +39 02896 33 456  
- Spain: +34 91 406 90 10  
- International distributors: +44 (0) 208 396 0080  
Hours:  
- United Kingdom: 9:00 A.M.–6:00 P.M. GMT, Monday-Friday except holidays  
- Mainland Europe: 9:00 A.M.–6:00 P.M. CET, Monday-Friday except holidays |
| **Asia Pacific**                    | E-mail: apsupport@microstrategy.com  
Web: https://support.microstrategy.com  
Fax: +81 3 5456 5464  
Phone:  
APAC (except Korea): +81 3 5456 5618  
Korea: +82 2 555 2525  
Hours: 9:00 A.M.–6:00 P.M. JST (Tokyo), Monday–Friday except holidays |
| **Latin America**                   | E-mail: latamsupport@microstrategy.com  
Web: https://support.microstrategy.com  
Fax: +55 11 3044 4088  
Phone: LATAM (except Argentina): +55 11 3054 1010  
Argentina: 0 800 444 MSTR  
Hours: 9:00 A.M.–6:00 P.M. (San Paulo), Monday–Friday except holidays |
Technical Support may be obtained by a Customer’s Support Liaisons. A "Support Liaison" is defined as a person whom the customer has designated as a point-of-contact with MicroStrategy’s support personnel. All customer inquiries and case communications must come through these named individuals. The customer may designate two employees to serve as their Support Liaisons. Customers may change their Support Liaisons two times per year, if necessary, so long as they provide written notice to MicroStrategy Technical Support of such change.

During the course of troubleshooting and researching issues, MicroStrategy Technical Support personnel may make recommendations that require administrative privileges on the MicroStrategy projects or that assume that the designated liaison has a security level that permits them to fully manipulate the MicroStrategy projects and has access to potentially sensitive project data such as security filter definitions. Although not a requirement, we recommend that customers only designate Support Liaisons who have permissions to be MicroStrategy project administrators. This will eliminate security conflicts and improve case resolution time.

When contacting MicroStrategy Technical Support, please provide the following information:

- name (first and last)
- company
- customer site (if different from company)
- phone and fax numbers
- e-mail address
- MicroStrategy software product(s) being used, including version number(s)
- error message(s)
- brief description of the case
- priority of the case
- steps taken to troubleshoot the case thus far
If the Support Liaison is unable to reach MicroStrategy Technical Support, the Support Liaison can leave a voice mail message or contact Technical Support via e-mail. The Support Liaison should include the following information in his/her message:

- name
- company
- brief description of the case
- preferred contact method and contact information

If this is your first call, you should also be prepared to provide the following:

- street address
- phone number
- fax number
- e-mail address

To help your Technical Support representative work with you to resolve the problem promptly and effectively, be prepared to provide the following additional information:

- issue number—please keep a record of the number assigned to each problem logged with MicroStrategy Technical Support, and be ready to provide it when inquiring about an existing issue
- software version and product registration numbers of the MicroStrategy software products you are using
• problem description:
  – What causes the condition to occur?
  – Does the condition occur sporadically or each time a certain action is performed?
  – Does the condition occur on all machines or just on one?
  – When did the condition first occur?
  – What events took place immediately prior to the first occurrence of the condition (for example, a major database load, a database move, a software upgrade)?
  – If there was an error message, what was its exact wording?
  – What steps have you taken to isolate and resolve the issue? What were the results?

• system configuration (the information needed for this purpose depends on the nature of the problem; not all items listed may be necessary):
  – computer hardware specifications (processor speed, RAM, disk space, and so on)
  – network protocol used
  – ODBC driver manufacturer and version
  – database gateway software version
  – (for MicroStrategy Web-related problems) browser manufacturer and version
  – (for MicroStrategy Web-related problems) Web server manufacturer and version

If the issue requires additional investigation or testing, you and the MicroStrategy Technical Support representative should agree on certain action items to be performed. You should perform any agreed-upon actions before contacting Technical Support again regarding the issue. If the Technical Support representative is responsible for an action item, you may call Technical Support at any time to inquire about the status of the issue.
Feedback

Please send any comments or suggestions about user documentation for MicroStrategy products to:

docfeedback@microstrategy.com

Send suggestions for product enhancements to:

support@microstrategy.com

When you provide feedback to us, please include the name and version of the products you are currently using. Your feedback is important to us as we prepare for future releases.
Introduction

Before you start reading this guide, you should review the concepts described in the *Installation and Configuration Guide*. This introduction summarizes the steps for setting up a project and explains the terms you should be familiar with before moving on to advanced reporting features.

You can also set up a project using the Project Creation Wizard, as described in Appendix G, *Project Creation Assistant*.

By the end of this chapter, you should understand what is involved in creating a report and how to proceed through the rest of this guide.
Basic MicroStrategy terminology

Source systems

_Source system_ refers to any system or file that captures or holds data of interest. This data is eventually analyzed in report format to determine answers to business-related questions.

The source system is the originating point of the data. For example, if you use your ATM card to conduct a transaction, the ATM is the point of transaction. It is the place where the data is gathered. In this example, the ATM gathers the information about how many dollars you deposited or withdrew from your account. The data is then written to a source system that is the large database behind the ATMs.

Deposits, withdrawals, sales transactions, inventory depletion, or replenishment are referred to as _transaction processing_. The source system records the transaction. Transaction processing data is usually stored in databases or mainframes for storage retrieval.

ETL process

The _extraction, transformation, and loading (ETL) process_ represents all of the steps necessary to move data from disparate source systems to an integrated data warehouse. The ETL tool is provided by a third-party vendor.

The first step is to extract or retrieve data from source systems. The second step is to transform the data and prepare it to be loaded into the data warehouse. Transformation procedures include converting datatypes and column names, eliminating bad data, correcting typographical errors, filling in incomplete data, and so on. The third and final step is to load the data into the warehouse.
The tools used to perform various aspects of the ETL process must contain information about the data, which facilitates the transfer of data from the source systems to the data warehouse. Specifically, such tools help you to

- store information about the structure and content of both the source system and data warehouse
- correlate the source system structure and content to that of the data warehouse

## Data warehouse

The *Installation and Configuration Guide* provides guidance on setting up a robust *data warehouse* environment. Data warehouses are generally based on some form of relational database and can be queried with Structured Query Language (SQL) to pull information from the warehouse into a report format. The data stored in the data warehouse originates from the source systems.

Data warehouses are designed and optimized for analytical processing. **Analytical processing** involves manipulating the data in the warehouse to calculate sales trends, growth patterns, trend reporting, profit analysis, and so on.

The data warehouse is a large database populated with data that is stored in tables. These databases have many tables, tracking many different pieces of information. It is not necessary to have multiple data warehouses as a data warehouse can store many databases and tables.

The data in the robust data warehouse can be populated with data from an existing source system using an ETL process. ETL takes the data from all sources, which can be spread over several different locations, and funnels them into one data warehouse.
Logical data model

The *logical data model* graphically represents the flow and structure of data in a business environment. It comprises facts, attributes, and hierarchies. Facts are numerical and aggregatable, such as daily revenue, inventory data, and hours worked. Once you have determined what your facts are, attributes allow you to answer the questions about a fact, such as a time frame for specific revenue totals. Hierarchies are groupings of attributes, ordered to reflect their relationship with other attributes. For example, you can group the attributes Year, Month, and Date to form the Time hierarchy.

Together, these three components—facts, attributes, and hierarchies—form your logical data model.

Physical warehouse schema

The *physical warehouse schema* is based on the logical data model. It is a detailed graphic representation of your business data. It organizes the logical model in a method that makes sense from a database perspective.

While the logical data model tells you what facts and attributes to create, the physical warehouse schema tells you where the underlying data for those objects is stored. The physical warehouse schema describes how your data is stored in the data warehouse.

Two key components make up the physical warehouse schema—tables and columns. Columns and tables in the physical warehouse schema represent facts and attributes from the logical data model. The rows in a table represent attribute elements and fact data.
Metadata database

Once you have a data warehouse populated with all the information your business needs to succeed, how do you retrieve the information from the database correctly and in the least amount of time?

The metadata contains information that facilitates the transfer of data between the data warehouse and the MicroStrategy application. It stores the object definitions and the information about the data warehouse, including its structure.

MicroStrategy uses the metadata to translate user requests into SQL queries and to translate the SQL queries back into MicroStrategy objects, such as reports.

The three types of objects that are stored in the metadata are:

- schema objects
- application objects
- configuration objects

Schema objects

Schema objects are created usually by a project designer. Schema objects relate the information in the logical data model and physical warehouse schema to the MicroStrategy environment. Facts, attributes, and hierarchies are examples of schema objects. These objects are developed in MicroStrategy Architect, which can be accessed from MicroStrategy Desktop.

Application objects

The report designer creates the application objects necessary to run reports. Application objects, which are developed in MicroStrategy Desktop and Web, are the building blocks for reports and documents. These application objects include reports, report templates, filters, metrics, prompts, and so on.
**Configuration objects**

Administrative and connectivity-related objects, also called *configuration objects*, are managed in MicroStrategy Server Administrator by an administrator role. Examples of configuration objects include users, groups, server definitions, and so on.

**Facts**

A *fact* has two characteristics: it is numerical and aggregatable. Examples of facts include revenue, inventory, and account balances.

There are some cases where a fact is not numerical or aggregatable, but these are rare.

Facts are stored in the data warehouse in fact tables. These fact tables comprise different columns, each cell representing a specific piece of information. Metrics, which are business measures, are created from this information.

SQL aggregations, such as `SUM` and `AVG`, are performed on the facts in the database tables. For example, in the following SQL statement, the `ORDER_AMT` column in the warehouse might correspond to the Order Amount fact in the MicroStrategy environment:

```
SELECT SUM(a21.ORDER_AMT) REGION
FROM ORDER_FACTa21
JOIN LU_EMPLOYEEa22
ON (a21.EMP_ID = a22.EMP_ID)
WHERE a22.CALL_CTR_ID in (5, 9, 12)
```

In this example, `ORDER_AMT` is the fact, whereas `SUM(a21.ORDER_AMT)` represents a metric.
Attributes

Once you have determined all the facts necessary to complete your business model, you identify the attributes for that model. **Attributes** act as holders of information, allowing you to add context to your facts in a report.

For example, if you had $10,000 in revenue, that number does not mean anything in a business sense unless you know the context, such as which region, the designated time frame for the sales, and what was the labor involved.

Simply put, attributes provide categories for the summarization of data.

Attribute elements

**Attribute elements** are the data shown on the report. Think of them as a sub-level of the attribute. For example, City might be the attribute, whereas London, Milan, and New York are the attribute elements.

In the data warehouse, attributes are usually represented by columns in a table, and attribute elements are represented by the rows.
Attribute relationships

Attribute relationships give meaning to the data in a logical data model by associating attributes based on business rules. The types of attribute relationships are

- one-to-one
- one-to-many
- many-to-many

One-to-one

In a one-to-one relationship, each element in the parent attribute is related to one and only one element in the child attribute, and each element in the child attribute is related to only one element in the parent.

In general, one-to-one relationships are modeled as attribute forms. Additional information on attribute forms can be found in Chapter 11, Attributes.

One-to-many

In a one-to-many relationship, each element in the parent attribute is related to more than one element in the child attribute, and each element in the child attribute is related to only one element in the parent.

For example, in a relationship between Year and Quarter, Year is the parent attribute and Quarter is the child attribute.
The following graphic depicts the relationship between the parent and child.

Many-to-many

In a \textit{many-to-many relationship}, each element in the parent attribute is related to more than one element in the child attribute, and each element in the child attribute is related to many elements in the parent.

In a car manufacturing plant, many models of cars are produced, and each comes in several colors. That is, there are many colors for a single type of car, and many types of cars can be associated with the same color.

Metrics

Metrics are analytical calculations performed against stored data (facts) to produce results that can then either be read as status material or analyzed for decision-making purposes. A metric can be defined within a report to specify the data to be
displayed in the report. This data can then be read or analyzed for decision-making purposes. Advanced metrics are discussed in Chapter 6, *Metrics*.

**Reports**

Once you have created your project, set up attributes and facts, and created a simple metric, you can run a report.

A report is a request for specific formatted data from the data warehouse. Reports can contain attributes and facts from the data warehouse, filters to determine how much data is used to generate the report, and metrics to perform calculations on the facts. More sophisticated functions, such as report limits, metric qualifications, dynamic aggregation, and shortcuts to reports and filters, allow you to create more functional and informative reports. These functions are described in the Reports chapter.

**Report Objects**

*Report Objects* are objects associated with a given report. At any time, a user can choose to view only a particular set of those objects. For example, you choose to show Metric1 and Metric2 but not Metric3 on the template.

Report Objects also indicate the lowest level of detail available in a report. This is accomplished by looking at the list of attributes in the Report Objects list. For example, a report with Year, Month, and Week in Report Objects has data for the metrics in that report at the Year, Month, and Week level. A user can choose to view only Year and Month on the template. In that case, the data is aggregated by default to the Month level, the lowest level of detail on the report.

The Reports Objects list is not shown by default in the Report Editor, although it is displayed on the left side of the Report Viewer. If it is not automatically displayed, choose **Show Report Objects** from the **View** menu.
Moving to advanced reporting

At this point you should have a project up and running, complete with some facts, attributes, and perhaps a simple metric or two.

You can now create a report with more detail, using the concepts described in this guide. You will learn how to

- create advanced facts
- create advanced attributes
- create nested and compound metrics
- apply advanced filters to your report
- manipulate the view definition and data definition of a report
- manipulate the hierarchy
- create a transformation
- create prompts and custom groups

You will also learn the following:

- usage of a data mart report
- customization of SQL statements
- advanced document layout
- usage of pass-through expressions
- partitioning of fact tables

Once you have understood and practised these concepts, you will be able to choose, manipulate, and format advanced reports that best answer your business questions.
Introduction

A report is a MicroStrategy object that represents a request for a specific set of formatted data from the data warehouse. Reports are the focus and goal of business intelligence. They allow users to gather business insight through data analysis.

The different parts of a report include: attributes and facts from the warehouse, filters that determine how much data is used to generate the report, and metrics to perform calculations on the facts. As you read through this chapter, you will learn about more sophisticated features that allow you to create more functional and informative reports.
Before you begin

The Reporting Essentials chapter of the Basic Reporting Guide contains fundamental information on report design. This advanced chapter builds on the concepts and procedures presented there by providing more technical details and advanced options for report design. Therefore, you should be familiar with the information from that chapter, such as Report Objects, Report Grid, Report Filter, and a general working knowledge of the Report Editor and its functions. A quick review of that chapter is included in the next section.

This chapter guides you through advanced reporting concepts in a hands-on way, although detailed instructions are not included. The online help contains such step-by-step procedures, if you need more guidance. The sample reports are intended to show you how reports are built and generated. After you read the chapter, explore the reports on your own to learn more and to better understand the concepts presented here.

The reports discussed in this chapter are saved in the MicroStrategy Tutorial. To simplify the content, the discussion and presentation of the reports are from Desktop only. However, you can access them from Web and perform many of the same operations. The directory path within Desktop is Public Objects\Reports\Technical Reports\Reports by Feature\Advanced Reporting Examples. You can follow the steps to interact with the reports, or you can view all of the sample reports without creating your own reports.

Remember to save any reports you create under a different name, so that you do not overwrite the sample reports in the MicroStrategy Tutorial.

Reporting Essentials review

The Reporting Essentials chapter of the Basic Reporting Guide provides an overview of the essential reporting topics you need to understand to begin building reports and creating a business intelligence application. These topics are explained in the following sections.
Report design versus report creation

*Report design* is the process of building reports from basic report components in MicroStrategy Desktop and Web. While report design is the most generic method for defining a report, it also requires the most in-depth knowledge of the project. In general, this method should be available only to the select group of advanced users and report designers who will design reports for others to use.

*Report creation* is the process of building reports from existing, predesigned reports either in Desktop or in Web. Given the wealth of reporting functionality that you can make available to your users, you have the ability to design reports that provide a wide range of options for users to create their own reports.

Report creation is different from report design in that it provides a more guided experience and does not require your users to have a thorough understanding of the project. This allows your users to create their own reports in a controlled, user-friendly environment.

Designing reports

You create reports in the Report Editor of Desktop, which has four report view modes:

- **Design View** describes the report definition and allows you to create and edit reports. The attributes, metrics, and other objects to be used in the report are displayed. You do not have to execute the report to view or modify the report structure.

- **Grid View** offers a formatted, cross-tabular display of the actual report data after the report is executed.

- **Graph View** is similar to Grid View, but the display is in a graphical format instead of cross-tabular.

- **SQL View** displays the SQL generated by the MicroStrategy Engine and executed in the warehouse. It also includes various execution statistics.
MicroStrategy Web provides the same report view modes, although the equivalent of SQL View is called Details.

You design reports in Design View, which allows you to select the metrics and attributes to use on the report. You can also define report filters, which determine the data used for metric calculation.

You can add various formatting options, such as fonts and styles, in either Design View or Grid View.

**Interactive report editing**

Once a report is saved, you have the option of allowing your users to edit it interactively while viewing the results without re-executing the report against the warehouse. This means that the changes are performed in Desktop or the Intelligence Server, rather than in the warehouse. The following functions are described fully in the *Reporting Essentials* chapter.

- Pivoting and page-by reorganizes report data by swapping objects within an axis or by moving objects from one axis to another.
- Sorting allows you to specify an ascending or descending order to present the report data for a particular row or column.
- The View filter restricts the amount of data displayed on the report, by controlling the subset of data to be displayed from the data retrieved from the database.
- Derived metrics are calculations defined on-the-fly with the data available in the report. They are based on existing metrics in the report to provide simple column math capability.
- Report Objects contain all of the objects available for display on the report. Use Report Objects to interactively modify the content of the report while viewing the report results. It displays the level of the report data definition. Data definition is discussed later in the chapter.
• Thresholds and stoplights allow you to highlight data that meets conditions by using different cell formats, symbols, and images or replacement text.

• Subtotals permit you to add, remove, and edit the subtotals at different levels for metrics on the report.

• Aliasing is the temporary renaming of objects for the report display.

• Outline Mode creates an indented grouping of related attribute elements, allowing you to expand and contract sections of related data.

• Exporting is rendering the report in different formats or applications, such as a spreadsheet or a word processor.

The basic report

The first report we will examine is a simple report. In Desktop, open the Basic Report from the MicroStrategy Tutorial, which displays the report in the Grid View, as shown below.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$ 514,524</td>
<td>$ 391,121</td>
<td>$123,403</td>
</tr>
<tr>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$ 325,888</td>
<td>$ 250,120</td>
<td>$ 79,768</td>
</tr>
<tr>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$ 390,888</td>
<td>$ 204,751</td>
<td>$ 95,137</td>
</tr>
<tr>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td>$316,786</td>
<td>$ 240,110</td>
<td>$ 76,676</td>
</tr>
<tr>
<td></td>
<td>Sondor</td>
<td>Melanie</td>
<td>$ 421,036</td>
<td>$ 313,975</td>
<td>$102,061</td>
</tr>
<tr>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$ 362,742</td>
<td>$ 275,208</td>
<td>$ 87,534</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Bernstein</td>
<td>Lawrence</td>
<td>$ 403,122</td>
<td>$ 305,730</td>
<td>$ 97,392</td>
</tr>
<tr>
<td></td>
<td>Brown</td>
<td>Vernon</td>
<td>$ 546,862</td>
<td>$ 415,009</td>
<td>$132,653</td>
</tr>
<tr>
<td></td>
<td>Corcoran</td>
<td>Peter</td>
<td>$ 346,565</td>
<td>$ 263,369</td>
<td>$ 83,196</td>
</tr>
<tr>
<td></td>
<td>Folks</td>
<td>Adrienne</td>
<td>$ 418,967</td>
<td>$ 317,132</td>
<td>$101,835</td>
</tr>
<tr>
<td></td>
<td>Hollywood</td>
<td>Robert</td>
<td>$ 591,776</td>
<td>$ 448,257</td>
<td>$143,519</td>
</tr>
<tr>
<td></td>
<td>Ingles</td>
<td>Walter</td>
<td>$ 377,583</td>
<td>$ 285,494</td>
<td>$ 91,089</td>
</tr>
<tr>
<td></td>
<td>Smith</td>
<td>Thomas</td>
<td>$ 420,931</td>
<td>$ 313,093</td>
<td>$102,048</td>
</tr>
<tr>
<td></td>
<td>Young</td>
<td>Sarah</td>
<td>$ 305,534</td>
<td>$ 231,887</td>
<td>$ 73,647</td>
</tr>
</tbody>
</table>

This report calculates the metrics Revenue, Cost, and Profit by the attributes of Region and Employee. Region and Employee define the level of the report, that is, the level at which the metrics are calculated.
Switch to Design View. The following screen shot displays the Basic Report in the Design View of Desktop. All of the panes mentioned in the next paragraphs have been opened and annotated for your use.

Select **Report Objects** from the **View** menu and notice that all the Report Objects are included in the grid. Recall that the Report Objects pane lists all of the objects for which data is retrieved from the database, as well as the derived metrics created for this report.

This report is used as a foundation to create the more advanced reports in this chapter.
Filtering

What is a filter?

A filter is used to select the data for calculating the metrics in the report. It also restricts the attribute elements included in the report. In our example, we use the Month filter, which does not allow April, May, and December data to be included in the metric calculations. For our purposes, these months are not representative of the normal business cycle, so the filter excludes them from calculations.

The Month filter is included in the Supporting Objects directory in the Advanced Reporting Examples folder.

Report filter example

Add the Month filter to the Basic Report, in Design View. For step-by-step instructions, refer to the online help. When you re-execute the report, it looks like the following

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>De Le Torre, Sandra</td>
<td>$354,350</td>
<td>$269,591</td>
<td>$84,759</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kelly, Laura</td>
<td>$215,628</td>
<td>$163,478</td>
<td>$52,150</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kiefferson, Jack</td>
<td>$229,233</td>
<td>$173,672</td>
<td>$55,561</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sawyer, Leanne</td>
<td>$198,976</td>
<td>$150,803</td>
<td>$48,173</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sonder, Melanie</td>
<td>$286,039</td>
<td>$215,651</td>
<td>$69,388</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yager, Beth</td>
<td>$220,460</td>
<td>$167,350</td>
<td>$53,110</td>
<td></td>
</tr>
</tbody>
</table>

If you do not want to create it yourself, this report is saved as Filter - Month Report Filter in the Tutorial.

Notice that the metrics have different values than in the Basic Report. For example, Leanne Sawyer’s contribution to revenue is $198,076. In the unfiltered report, her revenue was $316,786. In the Basic Report, all data for all months was retrieved from the data warehouse. The Revenue metric was calculated using all months. In this filtered report, April, May, and December amounts are not considered, so this metric does not include them in its calculations.
This is only one kind of filter; there are other types of filters too. Chapter 5, *Filters*, discusses filters in greater detail.

In summary, a filter affects the nature of the metric calculation by restricting the information used to compute the report metrics.

**What is a report limit?**

After all the metrics are calculated, you may need to further restrict the data, without changing how the calculations were performed. For example, you want to see only the top ten employees from a report that ranks all the employee sales. If you apply a report limit, the data used to calculate the sales rank is not affected.

A *report limit* specifies a set of criteria used to restrict the data returned in the report data set after the report metrics are calculated. Because it is based on the report’s metric values, a limit is applied after all of them are calculated.

The Report Editor allows you to set limits on any metric you want to apply to the report. Report limits are defined using operators such as between and less than. For more information on operators, see Appendix E, *Logical and Mathematical Operators for Filtering*.

**Report limit example**

Open the *Basic Report* again and note that the number of rows is 34. Add a report limit to the *Basic Report* by following the instructions that follow.

---

**To add a report limit**

1. Select *Report Data Options* from the Data menu.
2. Choose *Report Limit* under the Calculations folder.
3 Click **Modify** to access the Report Limit Editor.

4 Open the Sales Metrics folder. Double-click **Revenue**.

5 Change the Operator to be **Greater than**.

6 Enter a value of 320,000.

7 Click **Save and Close**.

8 Click **OK** to return to the report.

The report is redisplayed, as shown below.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$514,524</td>
<td>$391,121</td>
<td>$123,403</td>
</tr>
<tr>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$329,686</td>
<td>$250,120</td>
<td>$79,766</td>
</tr>
<tr>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$389,888</td>
<td>$294,751</td>
<td>$95,137</td>
</tr>
<tr>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$421,036</td>
<td>$313,975</td>
<td>$102,061</td>
</tr>
<tr>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$362,742</td>
<td>$275,206</td>
<td>$87,534</td>
</tr>
</tbody>
</table>

This report is saved as **Limit - Revenue > 320K** in the Tutorial.

The report limit restricts the report data to only those employees with revenue above $320,000. For example, Sawyer is included on the **Basic Report**, but because her revenue is only $316,786, she is not included on this report. Notice that the number of rows is now 32, where it was 34 before the report limit was applied.

The difference between report limits and report filters is very important, but it may still be a little hard to see in these examples. The next series of sample reports use other MicroStrategy functionality to further differentiate these two features.
The difference between report filters and report limits

Rank metrics apply a ranking number to the metric values for a given attribute. For an example, open the **Sales Rank report**. As shown in the following figure, this report is the Basic Report with two additional metrics—Revenue Rank and Revenue Rank (unfiltered).

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Revenue Rank</th>
<th>Revenue Rank (unfiltered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$ 514,524</td>
<td>$ 391,121</td>
<td>$123,403</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$ 329,888</td>
<td>$ 250,120</td>
<td>$ 79,768</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$ 389,888</td>
<td>$ 294,751</td>
<td>$ 95,137</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td>$ 316,786</td>
<td>$ 240,110</td>
<td>$ 76,676</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$ 421,036</td>
<td>$ 318,975</td>
<td>$102,061</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$ 362,742</td>
<td>$ 275,208</td>
<td>$ 87,534</td>
<td>28</td>
<td>28</td>
</tr>
</tbody>
</table>

These metrics rank employees based on the Revenue metric. The Revenue Rank metric uses the report filter in its calculation, while the Revenue Rank (unfiltered) metric ignores this report filter. This feature allows both filtered and unfiltered values on the same report. For example, when a filter for the Northeast region is added to the report, the calculation for Revenue Rank (the filtered metric) uses only the employees in that region. The unfiltered metric uses all the employees, regardless of region, to calculate its rank numbers. A complete example is provided in *Filtering with rank* below. Metric level filtering is also explained in more depth in Chapter 6, *Metrics*. In the report sample above, these two metrics display the same value because the report does not contain a filter.

**Sorting on rank**

To make the order of ranking easier to view, sort by the rank metric. In Grid View, right-click the Revenue Rank column and select **Sort rows by this column**. As you can see from the following report sample, the rows are re-arranged based on the value in the Revenue Rank column. The report data does not change, only its order on the report changes.
This report is saved as **Sort by Revenue Rank**.

### Filtering with rank

Switch to the Design View to add the Month filter to the sorted report. When you re-execute it, note the changed values in the Revenue Rank metric. In the following sample, the rankings that have changed are highlighted.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Revenue Rank</th>
<th>Revenue Rank (unfiltered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>Walker</td>
<td>Robert</td>
<td>$ 1,716,267</td>
<td>$ 1,301,142</td>
<td>$415,125</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Southwest</td>
<td>Bates</td>
<td>Michael</td>
<td>$ 783,886</td>
<td>$ 594,787</td>
<td>$189,099</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Northwest</td>
<td>Becker</td>
<td>Kyle</td>
<td>$ 692,441</td>
<td>$ 525,800</td>
<td>$166,641</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Southwest</td>
<td>Bell</td>
<td>Caitlin</td>
<td>$ 624,961</td>
<td>$ 473,529</td>
<td>$151,432</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Hollywood</td>
<td>Robert</td>
<td>$ 591,776</td>
<td>$ 448,257</td>
<td>$143,519</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Central</td>
<td>Gale</td>
<td>Loren</td>
<td>$ 575,268</td>
<td>$ 436,105</td>
<td>$139,163</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Southeast</td>
<td>McClain</td>
<td>Sean</td>
<td>$ 556,149</td>
<td>$ 420,844</td>
<td>$135,305</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Brown</td>
<td>Vernon</td>
<td>$ 548,862</td>
<td>$ 416,009</td>
<td>$132,853</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Southeast</td>
<td>Strome</td>
<td>Fred</td>
<td>$ 541,361</td>
<td>$ 409,968</td>
<td>$131,393</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Benner</td>
<td>Ian</td>
<td></td>
<td>$ 526,867</td>
<td>$ 399,590</td>
<td>$127,277</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

This report is saved as **Sort by Revenue Rank - Month Report Filter**.
Look at Ian Benner to understand what has occurred. In the previous report, his revenue was $526,867, which placed him as the tenth highest revenue producer. In this new report, his revenue is calculated at $393,866 because the report filter is applied before the metric value is determined. The revenue does not include April, May, and December. His new revenue rank is calculated as five, since the report filter affects the data used to calculate the Revenue metric. However, the Revenue Rank (unfiltered) metric still returns a ten because it is set to ignore the report filter.

**Report limits with rank**

Open the **Sort by Revenue Rank** report. Notice that the highest rank is 34 and there are 34 rows in the report. Now, add a report limit of revenue greater than $320,000, as described in the **Report limit example**. Re-execute the report to see the following results.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Revenue Rank</th>
<th>Revenue Rank (unfiltered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>Walker</td>
<td>Robert</td>
<td>$1,716,267</td>
<td>$1,301,142</td>
<td>$415,125</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Southwest</td>
<td>Bates</td>
<td>Michael</td>
<td>$783,886</td>
<td>$594,787</td>
<td>$189,099</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Becker</td>
<td>Kyle</td>
<td>$692,441</td>
<td>$525,800</td>
<td>$166,641</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Southwest</td>
<td>Bell</td>
<td>Caitlin</td>
<td>$624,961</td>
<td>$473,529</td>
<td>$151,432</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Hollywood</td>
<td>Robert</td>
<td>$591,776</td>
<td>$448,257</td>
<td>$143,519</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Central</td>
<td>Gale</td>
<td>Loren</td>
<td>$575,268</td>
<td>$436,105</td>
<td>$139,163</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Southeast</td>
<td>McClain</td>
<td>Sean</td>
<td>$556,149</td>
<td>$420,844</td>
<td>$135,305</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Brown</td>
<td>Vernon</td>
<td>$548,652</td>
<td>$416,009</td>
<td>$132,653</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Southeast</td>
<td>Strome</td>
<td>Fred</td>
<td>$541,361</td>
<td>$409,968</td>
<td>$131,393</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Benner</td>
<td>Ian</td>
<td></td>
<td>$526,866</td>
<td>$399,590</td>
<td>$127,277</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Central</td>
<td>Torrison</td>
<td>Mary</td>
<td>$333,377</td>
<td>$253,507</td>
<td>$79,870</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Northeast</td>
<td>Kelly</td>
<td>Laura</td>
<td>$329,888</td>
<td>$250,120</td>
<td>$79,768</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

This report is saved as **Sort by Revenue Rank - Report Limit - Revenue > 320K**.

Notice that the highest rank is now 32 and there are only 32 rows on the report. The last two rows from the previous report have disappeared because the revenue in each row was less than the report limit. None of the metrics changed values.
because a report limit does not affect how the metrics are calculated; the limit is applied at the level of the report after the metrics are calculated.

Simultaneous report filters and limits

Both report filters and report limits can be used on the same report because they are applied at different stages of the execution cycle.

Right-click the **Sort by Revenue Rank report** on the Desktop to open it in the Design View for editing. Add the Month filter as the report filter and a report limit of revenue greater than $320,000, as described previously. Execute the report. The results appear as displayed in the following figure.

This report is saved as **Sort by Revenue Rank - Report Filter & Report Limit**.

Notice that the report is much smaller than either the Sort by Revenue Rank - Month Filter report or the Sort by Revenue Rank - Limit - Revenue > 320K report. Only 15 rows are returned, as opposed to 34 or 32. Also notice that the Revenue, Cost, Profit, and Revenue Rank values are the same as the filtered report. However, the Revenue Rank (unfiltered) values are the same as the Revenue Rank - Limit. Why is this?
The first step in creating this report is calculating metrics. The data used in the metrics is restricted by the report filter, so information from April, May, and December is not included. All the metrics are calculated using this data, except for the unfiltered metric, which ignores the report filter. Its values are calculated on the full year’s worth of data.

The results after all the metric calculations are completed form the report data set. The report limit is applied to this data set. The employees with revenue less than $320,000 (the report limit) are removed from the display before the report is presented. Because the revenue is calculated on fewer months than the Revenue Rank - Month Filter report, more employees are discarded than from the previous limit.

In other words, the limit stays the same (greater than $320,000), but the filter changes the data considered in calculating each employee’s rank.

A report filter affects the data used to calculate metrics, whereas a report limit does not affect how the metrics are calculated. Report limits are applied at the level of the report after the metrics are calculated.

What is a metric qualification?

A metric qualification is a filtering condition based on the value of a metric. It contains an output level, which determines the level at which the metric is calculated and to which attributes the metric applies. Like every filter, a metric qualification changes the nature of the metric calculations, unlike a report limit, which is applied after the metrics are calculated.

Recall that the level of the Basic Report is Region and Employee—the attributes on the report. The output level of the metric qualification can remain at the report level, or it can be changed.

If the output level is the same as the report level, the results are usually the same as using a report limit. This is just a coincidence, however, because report limits and metric qualifications are calculated differently and at different times in the report execution cycle.
If the output level differs from the report level, the metrics are calculated at the output level. In the example that follows, the report level is region and employee. In the previous reports, the metrics were calculated for each employee using all brands and all products. When a metric qualification with an output level is applied to the report, the metrics are calculated with only the data that meets the metric qualification. Working through the sample report will help you better understand metric qualifications and output levels.

Whether or not the output level differs from the report level, a metric qualification affects the report data set. On the other hand, a report limit is applied after the metrics are calculated.

**Metric qualification example**

Right-click the **Sort by Revenue Rank report** on the Desktop and select **Edit** to edit the report. Add a metric qualification by following the steps that follow.

**To add a metric qualification**

1. Double-click in the Report Filter pane to add a qualification.

2. Select **Add a Set qualification** and click **OK**. A set qualification is based on a metric or attribute relationships.

3. Click the browse button next to **Output Level**.

4. Select **Calculate the output for the list of attributes**. This allows you to select the output level for the metric qualification.

5. Select **Brand** under the Products folder and click > to add it to the **Selected objects** list.

6. Click **OK**.

7. Click the browse button next to **Metric**.
8 Select Revenue in the Sales Metrics folder.

9 Click OK.

10 Keep the Function as Metric Value, but select Greater than from the Operator drop-down list.

11 Do not change Value, but type 320000 in the box next to it.

12 Click OK.

Execute the report. The results are displayed in the following figure.

This report is saved as Sort by Revenue Rank - Report Filter - Metric Qualification at the Brand Level.

The metric values on the report are different from those calculated for the Sort by Revenue Rank report. The Sort by Revenue Rank report produces values for each employee for all products. On the other hand, the metrics on this report are calculated only on those brands with revenue greater than $320,000 because of the metric qualification.

In the Sort by Revenue Rank report, Fred Strome’s revenue rank was nine, with revenue of $541,361. On this metric-qualified report, his revenue is $353,170, because any brands with revenue less than $320,000 were not included in the Revenue metric calculation. While his unfiltered revenue rank remains the same, he has moved up to eight in the revenue ranking. The unfiltered metric does not include the...
metric qualification, so it is calculated on all brands, and therefore, all products. In contrast, the metric qualification affects the other Rank metric, just as it affects the Revenue, Cost, and Profit metric calculations. That is, only brands with more than $320,000 of revenue are included in those calculations.

**An alternative explanation of metric qualification**

To help you understand metric qualification better, you can think of it as creating a temporary report. When the report is executed, the metric qualification first generates a temporary report in the background. In the earlier example, that report is a list of brands. The qualification is applied, so the report is trimmed to include only those brands with revenue in excess of $320,000. This report looks like the following.

<table>
<thead>
<tr>
<th>Brand</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sony</td>
<td></td>
<td>$2,509,622</td>
</tr>
<tr>
<td>Sharp</td>
<td></td>
<td>$2,253,700</td>
</tr>
<tr>
<td>Panasonic</td>
<td></td>
<td>$1,722,800</td>
</tr>
<tr>
<td>Harman Kardon</td>
<td></td>
<td>$908,200</td>
</tr>
<tr>
<td>Hitachi</td>
<td></td>
<td>$687,800</td>
</tr>
<tr>
<td>Minolta</td>
<td></td>
<td>$622,080</td>
</tr>
<tr>
<td>Hewlett Packard</td>
<td></td>
<td>$532,562</td>
</tr>
<tr>
<td>RCA</td>
<td></td>
<td>$528,530</td>
</tr>
<tr>
<td>Olympus</td>
<td></td>
<td>$454,400</td>
</tr>
<tr>
<td>Pioneer</td>
<td></td>
<td>$349,150</td>
</tr>
<tr>
<td>3Com</td>
<td></td>
<td>$340,940</td>
</tr>
</tbody>
</table>

This report is saved as **Revenue by Brand**.

Then this temporary report is applied to the actual report. Metrics are calculated including only those brands listed on the temporary report—Sony, Sharp, Panasonic, and so on. In essence, this report is the same as creating a filter for the set of brands Sony, Sharp, Panasonic, and so on. However, unlike that filter, the metric qualification is dynamically calculated based on the Revenue metric at the brand level. When new revenue data is added, the values can change.
In many cases, a report limit can generate more efficient SQL than a metric qualification. A metric qualification is contained in a separate pass of SQL, generating a temporary table at the output level. When this table is joined to the rest of the output, it limits the data included in the other metric calculations. Because it is another table, a metric qualification is a separate step in report execution. In contrast, a report limit is contained in a HAVING or WHERE clause in one of the final SQL passes. Therefore, using a report limit reduces the number of SQL passes needed to execute the report. However, since they often yield different results, do not choose a report qualification or a limit based solely on SQL efficiency.

What is report as filter?

Report as filter allows you to create a report and use it as a filter to generate another report. It is a different way to achieve the same results as a metric qualification, but it is easier to understand and create. Because the logic used to generate the final report is clearer, MicroStrategy recommends using it rather than the metric qualification.

In Desktop, you select Add a Shortcut to a Report to access the report as filter functionality.

Report as filter example

To create the same report as the metric qualification example, open the Sort by Revenue Rank report in the Report Editor. Add a new report filter. Select Add a Shortcut to a Report and choose the Revenue by Brand report. Execute the report. Sample report results are shown in the following figure.
This report is saved as **Sort by Revenue Rank - Report Filter - Report as Filter at the Brand Level.**

As with the metric qualification report, the metric values differ from the unfiltered Sort by Revenue Rank report. The values shown in the earlier figure are calculated only for the brands that are returned on the Revenue by Brand report chosen as the filter.

### Understanding how a report is executed

Now that you have designed reports containing filters and limits, you can better understand how a report is generated. The following table describes the steps to execute a report.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The objects in Report Objects and the report filter are used to calculate all the metrics, based on the data in the data warehouse.</td>
</tr>
<tr>
<td>2</td>
<td>A logical data set is generated in the database or brought back to the Intelligence Server. Optimally, the data set remains in the database to increase performance.</td>
</tr>
<tr>
<td>3</td>
<td>If there is a report limit, it is applied at the level of the Report Objects to further restrict the data set. The report limit is based on the result of the metric calculations from step 1.</td>
</tr>
<tr>
<td>4</td>
<td>If there are no other functions, the report is returned to the user and displayed in the selected format.</td>
</tr>
</tbody>
</table>
These four steps are the *data definition* section of the report execution. The data definition establishes how the data is accessed and manipulated in the data warehouse. Up to this point, this chapter has discussed data definition and the functionality that creates it.

The other functions noted in step 4 comprise the *view definition*, which represents how the data is viewed and manipulated in the Intelligence Server. The remainder of this chapter is concerned with manipulating the final report data set generated in step 3.

### Data definition and view definition objects

The following tables are samples of the information stored in the data definition and the view definition.

<table>
<thead>
<tr>
<th><strong>Data Definition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Report Filter</strong></td>
</tr>
<tr>
<td><strong>Report Objects</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Report Limits</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>View Definition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Grid</strong></td>
</tr>
<tr>
<td><strong>Formatting</strong></td>
</tr>
<tr>
<td><strong>Thresholds</strong></td>
</tr>
<tr>
<td><strong>View Filter</strong></td>
</tr>
<tr>
<td><strong>Derived Metrics</strong></td>
</tr>
<tr>
<td><strong>Subtotals</strong></td>
</tr>
<tr>
<td><strong>Sorting</strong></td>
</tr>
</tbody>
</table>
Report designers are generally concerned with data definition while report creators usually focus on view definition. Report designers work on the details of reports to create a context or environment for the report creators to work in. This environment allows report creators to work within defined limits, ensuring that only reasonable queries are submitted to the database. Reasonable means that irrelevant data sets cannot be created, nor can huge amounts of data be retrieved from the warehouse. This designer-versus-creator convention allows a group of report designers to be trained about more advanced report functions while report creators can manipulate reports without needing to understand the report execution details. Through privileges, you can assign different levels of functionality to different users. More information on privileges is included later in this chapter.

**Intelligent Cubes**

The *Intelligent Cube* takes advantage of the separation of the data definition and the view definition. The cube is a shared copy of the report data saved in memory and is used for manipulation of the view definition. The division allows multiple reports with different views to share a common data definition. This division allows the Analytical Engine to perform highly interactive analysis against a set of data without accessing the data warehouse. In practical terms, you can modify reports and navigate through the data without leaving the executed report or waiting for an additional execution against the data warehouse.

The following diagram represents the Intelligent Cube, different views that can use the Intelligent Cube, and its underlying report cache.
The report view is an in-memory representation of the current view of a report, based on the view definition of that report. Each user running the same report has a unique report view on the Intelligence Server. Manipulating the report views is done after the report has been executed and uses Intelligent Cube technology. Intelligent Cubes are automatically instantiated whenever a report is executed; you do not have to manually create Intelligent Cubes.

The MicroStrategy Intelligence Server leverages the Intelligent Cube technology for in-memory report manipulation, such as formatting and sorting. You can exploit Intelligent Cubes when you design reports by allowing report creators to use Intelligent Cubes when they manipulate the view definition of reports.

For the purposes of MicroStrategy, the data definition is equivalent to the intelligent cube.
The report cache is created by an Intelligence Server schedule based on the data definition and the caching properties of the report. It contains pre-processed report data and is stored on disk. The Intelligent Cube is identical to the report cache but is stored in the Intelligence Server memory. However, the Intelligence Server uses sophisticated memory management rules to decide if and when to move an Intelligent Cube to disk.

The Analytical Engine has been enhanced to use Intelligent Cubes to allow manipulation of the data displayed in the report view.

The next sections of this chapter discuss view definition topics.

**View filters**

**What is a view filter?**

A view filter is a quick qualification applied in memory to the report data set. Because it affects only the view definition, the report does not have to be re-executed in the data warehouse. A view filter works in a manner similar to sorting and subtotals, with which you are already familiar.

A view filter, just as the report limit, is always applied at the level of the Report Objects list. However, the report limit and the view filter are not interchangeable. A report limit restricts the size of the report data set returned from the data warehouse. In contrast, the view filter is applied to the report data set without altering its size, allowing you to view a subset of that information. It retrieves the information quickly because a view filter dynamically accesses data already in memory.
When considering how to build a report, a report designer must balance the memory usage and the processing power of the data warehouse and the Intelligence Server. A report limit is more efficient in terms of data size because it does not return unnecessary information from the data warehouse. However, the data that a view filter does not display may not be unnecessary, but rather the user does not need to display it currently. He may need to display it later.

If a report limit is too restrictive, its utility is reduced because users must more frequently redefine their data definition to find the information they want. A view filter is more flexible, allowing users to refine the analysis after the report is executed, but it is more demanding on the Intelligence Server. A view filter is intended to be a post-report execution function to provide further investigation and refinement of the report data set. As such, it is aimed at a report creator, whereas the report limit is usually used by the report designer.

A report designer must consider the following:

<table>
<thead>
<tr>
<th></th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>View filter</strong></td>
<td>Flexible:</td>
<td>Less efficient:</td>
</tr>
<tr>
<td></td>
<td>• More information is available to allow further analysis.</td>
<td>• Intelligence Server must perform more work.</td>
</tr>
<tr>
<td></td>
<td>• Attributes can be used.</td>
<td>• More memory is used.</td>
</tr>
<tr>
<td><strong>Report limit</strong></td>
<td>Efficient:</td>
<td>Less flexible:</td>
</tr>
<tr>
<td></td>
<td>• Less information is returned from the data warehouse.</td>
<td>• Further analysis may require more data from the data warehouse.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Only metrics can be used as qualifiers.</td>
</tr>
</tbody>
</table>

MicroStrategy provides the flexibility to choose report filters, report limits, as well as view filters on a report-by-report basis. Each has a different role in the execution and business meaning of a report.
View filter example

Open the **Basic Report**. To concentrate on only a few employees without creating a new report, you can apply a view filter.

---

To create a view filter

1. Click in the View Filter pane to start a new qualification.

2. From the **Field** drop-down list, select **Employee**, then **Choose from a list**.

   The Field drop-down list only includes Report Objects. When you work with report limits or filters, you can choose any object, even if it is not on the report.

3. From the **Operator** drop-down list, select **In list**.

4. From the **Value** drop-down list, choose **Select Elements**.

5. Double-click on the following names:
   - Bell
   - Benner
   - Conner
   - Johnson
   - Kelly
   - Kieferson

6. Click **OK**.
When the report is redisplayed, the report looks like the following:

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Kelly</td>
<td>Laura</td>
<td>$329,888</td>
<td>$250,120</td>
<td>$79,768</td>
</tr>
<tr>
<td></td>
<td>Kieffer</td>
<td>Jack</td>
<td>$369,888</td>
<td>$264,751</td>
<td>$95,137</td>
</tr>
<tr>
<td>Southeast</td>
<td>Benner</td>
<td>Ian</td>
<td>$525,867</td>
<td>$399,590</td>
<td>$127,277</td>
</tr>
<tr>
<td>South</td>
<td>Conner</td>
<td>Beatrice</td>
<td>$380,503</td>
<td>$287,582</td>
<td>$92,921</td>
</tr>
<tr>
<td>Southwest</td>
<td>Bell</td>
<td>Caitlin</td>
<td>$524,961</td>
<td>$473,529</td>
<td>$151,432</td>
</tr>
<tr>
<td></td>
<td>Johnson</td>
<td>Andrew</td>
<td>$474,239</td>
<td>$360,044</td>
<td>$114,195</td>
</tr>
</tbody>
</table>

The report is saved as View Filter.

The only employees displayed are those in the list you created, for a total of six rows. Notice that the metrics are calculated at the lowest level of the Report Objects, regardless of what is on the grid when the view filter is applied.

To return to the original report, click Clear in the View Filter pane.

## Derived metrics

### What is a derived metric?

While viewing a report, you may want to perform calculations between columns. For example, a quick comparison, such as Metric1 - Metric2, may be useful. Derived metrics allow you to perform such column math, or math between metrics, in the report. A derived metric is a calculation based on the metrics already in the report. Derived metrics are generated based on the report data set.

Derived metrics are always evaluated in memory, so they do not need to access the data warehouse. Although they only present the data already available on the report in a different way, they are a powerful and easy-to-use tool. For example, you can use derived metrics to quickly perform on-the-fly analyses such as margins, contributions, and differences between metrics already on the report.
**Derived metric example**

Open the **Basic Report** again. To quickly compare the revenue values, you want to round them up to the thousandths place.

---

**To create a derived metric**

1. From the Insert menu, select **New Metric**. Notice that you can only choose Report Objects or functions and operators, that is, objects already on the report.

2. Double-click **Revenue**.

3. Type \( \frac{1}{1000} \) in the definition pane.

4. In the **Name** field above the formula, replace the default name “New Metric” with **Derived Revenue (K)**.

5. Click **OK**.

6. Select **Derived Revenue (K)** on the grid and drag it to the right.

7. To format the results, use the Formatting toolbar:
   - Select **Derived Revenue (K)** for the section and **Values** for the subsection.
   - Click **B** to bold the values.
   - Click **$** to format the values as currency.
   - Click the **Decrease the decimal** icon twice to remove cents from the display.
This report is saved as Derived Metrics.

Since the new column is only a different representation of data already on the report, the report does not have to be rerun against the data warehouse.

For more information on derived metrics, particularly the syntax, refer to Chapter 6, Metrics.

## Dynamic aggregation

### What is dynamic aggregation?

Sometimes you want to change the level of report aggregation on the fly, while you are reviewing the report data. For example, the Basic Report is calculated at the region and employee level. To see the Revenue, Cost, and Profit values at the region level, you can roll up the report data set from employee to region without accessing the data warehouse.
Dynamic aggregation occurs when you move an attribute from the grid to Report Objects. The metric values roll up to the new level of the grid. Dynamic aggregation occurs whenever the attributes in Report Objects are not the same as the attributes on the grid. Dynamic aggregation happens on the fly, in memory.

The Analytical Engine selects the best aggregation function to use, by default. However, you can also specify the function for each metric. You can use any of the standard predefined subtotal functions or the user-defined subtotals. For more information on this setting, see the Dynamic aggregation section in Chapter 6, Metrics.

Not all metrics can be rolled up. For more information, see Exceptions to dynamic aggregation in this chapter.

Dynamic aggregation example

Triggering dynamic aggregation achieves the same effects as adding subtotals to a report. To demonstrate this, add subtotals to the Basic Report, remove them, and then move attributes to Report Objects to cause dynamic aggregation. Both functions display the same values.

Open the Basic Report, which is displayed at the region and employee level, and add subtotals to view the regional totals. From the Data menu, choose Subtotals, then select Total. Note that Revenue for the Northeast is $2,334,864. Remove the subtotals.

To roll the metrics up to the region level, select Employee in the grid and drag it to Report Objects. Notice that Employee in Report Objects is no longer bold, representing that the attribute is not included on the grid. The report is redisplayed showing regional values only. Just as on the subtotalled report, Northeast Revenue is $2,334,864.
This report is saved as **Dynamic Aggregation**.

To obtain the new values, the metrics are recalculated in memory at the regional level. You can easily return to the employee level by restoring the Employee attribute to the grid.

You can also move metrics from the grid to Report Objects. However, moving a metric does not affect the level of the report; so it does not trigger dynamic aggregation. Instead, the metric is simply hidden from view.

### View filter effects

#### The effect of view filters on metrics

When a report is executed, the metrics are calculated to create the report data set. The view filter restricts the rows of the report data set that are displayed. Derived metrics are then computed based on the report data set. In other words, since the view filter is applied before derived metrics are calculated, the results change if the view filter alters the data considered for calculation of the derived metrics.

<table>
<thead>
<tr>
<th>Region</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$2,334,864</td>
<td>$1,770,285</td>
<td>$564,579</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>$3,413,340</td>
<td>$2,586,971</td>
<td>$826,369</td>
</tr>
<tr>
<td>Southeast</td>
<td>$2,016,186</td>
<td>$1,526,932</td>
<td>$489,254</td>
</tr>
<tr>
<td>Central</td>
<td>$1,773,270</td>
<td>$1,344,541</td>
<td>$428,729</td>
</tr>
<tr>
<td>South</td>
<td>$1,380,991</td>
<td>$1,045,658</td>
<td>$335,333</td>
</tr>
<tr>
<td>Northwest</td>
<td>$1,485,182</td>
<td>$1,126,729</td>
<td>$358,453</td>
</tr>
<tr>
<td>Southwest</td>
<td>$2,816,334</td>
<td>$2,135,068</td>
<td>$681,266</td>
</tr>
<tr>
<td>Web</td>
<td>$1,716,267</td>
<td>$1,301,142</td>
<td>$415,125</td>
</tr>
</tbody>
</table>
**Metric and view filter example**

Open the **Derived Metrics** report and add the view filter described in the *View filter example*.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Derived Revenue (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Kelly</td>
<td>Laura</td>
<td>$329,388</td>
<td>$250,120</td>
<td>$79,768</td>
<td>$330</td>
</tr>
<tr>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$389,888</td>
<td>$294,751</td>
<td>$95,137</td>
<td>$390</td>
</tr>
<tr>
<td>Southeast</td>
<td>Berner</td>
<td>Ian</td>
<td>$526,867</td>
<td>$399,590</td>
<td>$127,277</td>
<td>$527</td>
</tr>
<tr>
<td>South</td>
<td>Corner</td>
<td>Beatrice</td>
<td>$380,503</td>
<td>$287,582</td>
<td>$92,921</td>
<td>$381</td>
</tr>
<tr>
<td>Southwest</td>
<td>Bell</td>
<td>Caitlin</td>
<td>$624,961</td>
<td>$473,529</td>
<td>$151,432</td>
<td>$625</td>
</tr>
<tr>
<td></td>
<td>Johnson</td>
<td>Andrew</td>
<td>$474,239</td>
<td>$360,044</td>
<td>$114,195</td>
<td>$474</td>
</tr>
</tbody>
</table>

This report is saved as **View Filter- Metrics**.

The report is reduced in size, including only those employees selected in the view filter. This view filter is applied at the level of the Report Objects, and all of the attributes still appear on the report. None of the metrics, including the derived metric, change their value. If an attribute is moved from the grid to Report Objects, the values change, as described in the next section.

**The effect of view filters on dynamic aggregation**

When a report with a view filter is rolled up, only the data that meets the filter criteria is considered in the metric values. The metrics are recalculated at the new level of the report.
Dynamic aggregation and view filter example

Open the View Filter report. Drag Employee from the grid to Report Objects. The results are:

<table>
<thead>
<tr>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td></td>
<td>$719,776</td>
<td>$544,871</td>
<td>$174,905</td>
</tr>
<tr>
<td>Southeast</td>
<td></td>
<td>$526,867</td>
<td>$399,590</td>
<td>$127,277</td>
</tr>
<tr>
<td>South</td>
<td></td>
<td>$380,503</td>
<td>$287,582</td>
<td>$92,921</td>
</tr>
<tr>
<td>Southwest</td>
<td></td>
<td>$1,099,200</td>
<td>$833,573</td>
<td>$265,627</td>
</tr>
</tbody>
</table>

This report is saved as View Filter - Dynamic Aggregation.

Dynamic aggregation occurs because the attributes on Report Objects no longer match the attributes on the grid. The employees selected in the view filter do not belong to the Mid-Atlantic, Central, Northwest, or Web regions, so these regions are not displayed. Now that the report data set is restricted and rolled up, the metric values include only the employees in the view filter.

To understand the effects of a view filter on dynamic aggregation better, compare the following reports. This sample of the Basic Report displays the revenue for each employee in the Northeast and Mid-Atlantic regions.

Although other regions are included on the report, they are not all displayed in the samples due to space constraints.

When Employee is moved to Report Objects, dynamic aggregation occurs, and the revenue is rolled up to the region level. This is illustrated in the second report. Add the view filter from the View filter example to the Basic Report to obtain the third report. The view filter does not include any employees from the Mid-Atlantic region; so that region is no longer displayed on the report. When Employee is moved to Report Objects on this report, the revenue is again rolled up to the region level, as shown in the fourth report. The revenue values between the two rolled-up reports differ, because the Revenue metric in the filtered report includes only the revenue from the employees in the view filter.
# 1: Basic report

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$ 514,524</td>
</tr>
<tr>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$ 329,888</td>
</tr>
<tr>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$ 369,999</td>
</tr>
<tr>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td>$ 316,786</td>
</tr>
<tr>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$ 421,036</td>
</tr>
<tr>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$ 362,742</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Bernstein</td>
<td>Lawrence</td>
<td>$ 403,122</td>
</tr>
<tr>
<td></td>
<td>Brown</td>
<td>Vernon</td>
<td>$ 548,862</td>
</tr>
<tr>
<td></td>
<td>Corcoran</td>
<td>Peter</td>
<td>$ 346,565</td>
</tr>
<tr>
<td></td>
<td>Folks</td>
<td>Adrienne</td>
<td>$ 410,967</td>
</tr>
<tr>
<td></td>
<td>Hollywood</td>
<td>Robert</td>
<td>$ 591,776</td>
</tr>
<tr>
<td></td>
<td>Ingles</td>
<td>Walter</td>
<td>$ 377,583</td>
</tr>
<tr>
<td></td>
<td>Smith</td>
<td>Thomas</td>
<td>$ 420,931</td>
</tr>
<tr>
<td></td>
<td>Young</td>
<td>Sarah</td>
<td>$ 305,534</td>
</tr>
</tbody>
</table>

#2: Basic report rolled up to Region

<table>
<thead>
<tr>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$ 2,334,864</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>$ 3,413,340</td>
<td></td>
</tr>
<tr>
<td>Southeast</td>
<td>$ 2,016,186</td>
<td></td>
</tr>
</tbody>
</table>

#3: Basic report + view filter

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Kelly</td>
<td>Laura</td>
<td>$ 329,888</td>
</tr>
<tr>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$ 389,888</td>
</tr>
<tr>
<td>Southeast</td>
<td>Benner</td>
<td>Ian</td>
<td>$ 526,867</td>
</tr>
</tbody>
</table>

#4: Basic report + view filter, rolled up to Region

<table>
<thead>
<tr>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$ 719,776</td>
<td></td>
</tr>
<tr>
<td>Southeast</td>
<td>$ 526,867</td>
<td></td>
</tr>
</tbody>
</table>

Metric, view filter, and dynamic aggregation example

When an attribute is moved to Report Objects, dynamic aggregation occurs. The Analytical Engine aggregates the metrics as accurately as possible in memory. Sometimes, as in count distinct metrics, aggregation is not possible, resulting in dashes that signifies the inability to calculate results. As always, derived metrics are recalculated based on the data in the filtered grid.
On the report you created in the previous example (Dynamic aggregation and view filter example), you can roll the values up to the region level. You can remove Employee from the report grid to accomplish this dynamic aggregation. First, though, put subtotals on the report, to verify the dynamic aggregation totals. Notice that the derived revenue for the Northeast is $720 and for Southeast is $527. Remove the subtotals.

Now, roll up the values by selecting Employee on the grid and dragging it to Report Objects. The report is redisplayed showing regional values only. The Derived Revenue metric is again $720.

<table>
<thead>
<tr>
<th>Region</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Derived Revenue (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$719,776</td>
<td>$544,871</td>
<td>$174,905</td>
<td>$720</td>
</tr>
<tr>
<td>Southeast</td>
<td>$526,867</td>
<td>$399,590</td>
<td>$127,277</td>
<td>$527</td>
</tr>
<tr>
<td>South</td>
<td>$390,503</td>
<td>$267,582</td>
<td>$92,921</td>
<td>$301</td>
</tr>
<tr>
<td>Southwest</td>
<td>$1,099,200</td>
<td>$833,573</td>
<td>$265,627</td>
<td>$1,099</td>
</tr>
</tbody>
</table>

This report is saved as View Filter - Metrics - Dynamic Aggregation.

All of the metric values—Revenue, Cost, Profit, and Derived Revenue—are evaluated at the region level, but only for the employees in the view filter.

**Metric qualification in the view filter**

If you include a metric qualification in a view filter, it is applied at the level of the Report Objects. In the previous sections, you learned that a metric qualification returns a set of attribute elements that is applied as a filter. A metric qualification works in the same way when included in a view filter.
Metric qualification in the view filter example

Open the Derived Metrics report, which contains Revenue, Cost, and Profit metrics plus the Derived Revenue metric. Notice this report has 34 rows and the first employee listed is De Le Torre, with revenue of $514,524. Notice also that Kelly’s revenue is $329,888.

Now, add a metric qualification in the view filter. Set the view filter to revenue less than $500,000. Remember to click Apply to view the results, which are shown in the following figure.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Derived Revenue (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Kelly</td>
<td>Laura</td>
<td>$329,888</td>
<td>$250,120</td>
<td>$79,768</td>
</tr>
<tr>
<td></td>
<td>Kiefer</td>
<td>Jack</td>
<td>$389,898</td>
<td>$294,751</td>
<td>$95,137</td>
</tr>
<tr>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td>$316,766</td>
<td>$240,110</td>
<td>$76,656</td>
</tr>
<tr>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$421,036</td>
<td>$310,975</td>
<td>$102,061</td>
</tr>
<tr>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$362,742</td>
<td>$275,208</td>
<td>$87,534</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Bernstein</td>
<td>Lawrence</td>
<td>$403,122</td>
<td>$305,730</td>
<td>$97,392</td>
</tr>
<tr>
<td></td>
<td>Corcoran</td>
<td>Peter</td>
<td>$346,565</td>
<td>$263,369</td>
<td>$83,196</td>
</tr>
<tr>
<td></td>
<td>Folks</td>
<td>Adrienne</td>
<td>$418,967</td>
<td>$317,132</td>
<td>$101,835</td>
</tr>
<tr>
<td></td>
<td>Ingles</td>
<td>Walter</td>
<td>$377,593</td>
<td>$206,494</td>
<td>$91,089</td>
</tr>
<tr>
<td></td>
<td>Smith</td>
<td>Thomas</td>
<td>$420,931</td>
<td>$318,093</td>
<td>$102,838</td>
</tr>
<tr>
<td></td>
<td>Young</td>
<td>Sarah</td>
<td>$305,534</td>
<td>$231,887</td>
<td>$73,647</td>
</tr>
</tbody>
</table>

This report is saved as View Filter - Metric Qualification.

The metric qualification is calculated at the report level, which is employee. Therefore, the report omits employees whose revenue is greater than $500,000. The report displays only 21 rows and the details of the employee De Le Torre are not displayed.

Dynamic aggregation with a metric qualification in the view filter example

When the data in a report is rolled up to a new level, the same logic applies. That is, the metric qualification provides a set of attribute elements that are used to filter the report data set before the metrics are calculated at the higher level.
On the report you created for the *Metric qualification in the view filter example*, move **Employee** from the grid to Report Objects. This rolls the report up to the region level, as shown below.

<table>
<thead>
<tr>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Derived Revenue (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$1,820,340</td>
<td>$1,379,164</td>
<td>$441,176</td>
<td>$1,820</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>$2,272,702</td>
<td>$1,722,705</td>
<td>$549,997</td>
<td>$2,273</td>
<td></td>
</tr>
<tr>
<td>Southeast</td>
<td>$391,009</td>
<td>$296,530</td>
<td>$95,279</td>
<td>$392</td>
<td></td>
</tr>
<tr>
<td>Central</td>
<td>$1,196,002</td>
<td>$938,435</td>
<td>$269,566</td>
<td>$1,198</td>
<td></td>
</tr>
<tr>
<td>South</td>
<td>$867,240</td>
<td>$655,983</td>
<td>$211,257</td>
<td>$867</td>
<td></td>
</tr>
<tr>
<td>Northwest</td>
<td>$792,741</td>
<td>$630,929</td>
<td>$161,812</td>
<td>$793</td>
<td></td>
</tr>
<tr>
<td>Southwest</td>
<td>$692,654</td>
<td>$676,056</td>
<td>$215,798</td>
<td>$693</td>
<td></td>
</tr>
</tbody>
</table>

This report is saved as **View Filter - Metric Qualification - Dynamic Aggregation**.

Only those employees who meet the metric qualification at the employee level are included when the metrics are rolled up to the region level. If you are unsure of the level of the metric qualification, click the i at the end of the View Filter definition, as shown in the following figure.
View definition in the report execution cycle

Now that you have designed reports containing view definition objects, we can revisit the report execution cycle. The following table includes both the data definition, which also defines the Intelligent Cube, and view definition steps to execute a report.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The objects in Report Objects and the report filter are used to calculate all the metrics, based on the data in the data warehouse.</td>
</tr>
<tr>
<td>2</td>
<td>A logical data set is generated in the database or brought back to the Intelligence Server. Optimally, the data set remains in the database to increase performance.</td>
</tr>
<tr>
<td>3</td>
<td>If there is a report limit, it is applied at the level of the Report Objects to further restrict the data set. The report limit is based on the result of the metric calculations from step 1.</td>
</tr>
<tr>
<td>4</td>
<td>If a view filter exists, it is applied.</td>
</tr>
<tr>
<td>5</td>
<td>Derived metrics are calculated. If any attributes are moved from the grid to Report Objects, the data is rolled up to the level of the attributes on the grid.</td>
</tr>
<tr>
<td>6</td>
<td>The report is returned to the user and displayed in the selected format.</td>
</tr>
</tbody>
</table>

The first three steps are the data definition, which are interactions with the data warehouse. Steps 4 through 6 are concerned with the view definition, and they take place in memory.
Exceptions to dynamic aggregation

The ability to roll up data in memory is useful for quick report interaction and analysis. However, not all metrics can be rolled up with an additional aggregation function. Instead, if the data is required at the higher level, it first must be recalculated from the detail data available only in the data warehouse.

For example, a count distinct tallies each different item only once, as opposed to a regular count, which adds up all the items. For example, if employee A sells four widgets and two gizmos, a count of items sold returns six. The count distinct of items sold is two. Employee B sells ten widgets and no gizmos, so his count is ten and count distinct is one. For example, to aggregate the data at a level higher than employee, remove Employee from the grid to display the data at the regional level. The counts are added together, for a total of sixteen. Adding the count distinct, however, would incorrectly return three. The only items sold were widgets and gizmos, so the proper answer is two. Note that the correct answer can only be obtained by accessing the lowest level of detail in the data warehouse.

To create a count distinct metric, use the Insert Function Wizard in the Metric Editor. This wizard displays the parameters of a function, so you can easily select Distinct. For more information, see the online help about the Insert Function Wizard.

For those metrics that can be rolled up, you can specify which function to use. On the Subtotals/Aggregation tab in the Metric Editor, change the Dynamic Aggregation Function from Default. For more information, see the Dynamic aggregation section in Chapter 6, Metrics.
Dynamic aggregation exception example

Open the **Dynamic Aggregation - Region - Employee** report, which is displayed below.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Standard Deviation of Revenue</th>
<th>Max Revenue</th>
<th>Min Revenue</th>
<th>Count Distinct (Items Sold)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$514,524</td>
<td>$2,586.65</td>
<td>$22,000.00</td>
<td>$5.00</td>
<td>331</td>
</tr>
<tr>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$329,888</td>
<td>$1,616.65</td>
<td>$12,600.00</td>
<td>$3.00</td>
<td>315</td>
</tr>
<tr>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$389,888</td>
<td>$2,211.07</td>
<td>$18,000.00</td>
<td>$5.00</td>
<td>322</td>
</tr>
<tr>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td>$316,786</td>
<td>$1,735.34</td>
<td>$17,100.00</td>
<td>$5.00</td>
<td>323</td>
</tr>
<tr>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$421,036</td>
<td>$2,402.56</td>
<td>$19,000.00</td>
<td>$6.00</td>
<td>326</td>
</tr>
<tr>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$362,742</td>
<td>$1,853.01</td>
<td>$13,200.00</td>
<td>$3.00</td>
<td>323</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Bernstein</td>
<td>Lawrence</td>
<td>$403,122</td>
<td>$2,221.14</td>
<td>$18,900.00</td>
<td>$6.00</td>
<td>329</td>
</tr>
<tr>
<td></td>
<td>Brown</td>
<td>Vernon</td>
<td>$548,862</td>
<td>$2,906.46</td>
<td>$27,900.00</td>
<td>$3.00</td>
<td>333</td>
</tr>
<tr>
<td></td>
<td>Corcoran</td>
<td>Peter</td>
<td>$346,565</td>
<td>$2,799.98</td>
<td>$14,000.00</td>
<td>$3.00</td>
<td>312</td>
</tr>
<tr>
<td></td>
<td>Folks</td>
<td>Adrienne</td>
<td>$418,967</td>
<td>$2,242.77</td>
<td>$18,000.00</td>
<td>$5.00</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>Hollywood</td>
<td>Robert</td>
<td>$591,776</td>
<td>$3,223.10</td>
<td>$27,000.00</td>
<td>$5.00</td>
<td>332</td>
</tr>
<tr>
<td></td>
<td>Ingles</td>
<td>Walter</td>
<td>$377,583</td>
<td>$2,077.28</td>
<td>$18,900.00</td>
<td>$3.00</td>
<td>326</td>
</tr>
<tr>
<td></td>
<td>Smith</td>
<td>Thomas</td>
<td>$420,931</td>
<td>$2,317.84</td>
<td>$16,000.00</td>
<td>$3.00</td>
<td>323</td>
</tr>
<tr>
<td></td>
<td>Young</td>
<td>Sarah</td>
<td>$305,534</td>
<td>$1,577.30</td>
<td>$13,000.00</td>
<td>$5.00</td>
<td>316</td>
</tr>
</tbody>
</table>

This report is a basic analysis of revenue at the employee level. It includes the revenue for each employee, and then the standard deviation, maximum, and minimum revenue calculated for each employee. The final metric is a count of distinct items sold by that employee. A count distinct means that each item is counted only once, as opposed to a regular count which adds up how many items the employee sold.

To roll this report up to the region level, move **Employee** from the grid to Report Objects. The results of the aggregation are shown in the next report sample.
This report is saved as **Dynamic Aggregation - Region**.

Revenue can be rolled up, as the sum of all employees in the region. Standard deviation cannot be merely summed; it must be recalculated at the region level. The report does not contain this information, so dashes are displayed in the standard deviation column.

The minimum and maximum values can be calculated at the region level, because all the needed information is contained in the report. Count distinct cannot be rolled up, because duplicate items can exist between employees, and therefore a sum will not be valid.

If you need to calculate the values, you can completely remove Employee from the report, not just the grid. Simply right-click **Employee** in the Report Objects pane and select **Remove from report**. A warning appears, because the report must be regenerated to return the correct answers. Therefore, this action is no longer a function of the view definition, but instead, the data definition. The results are shown in the next report sample.

<table>
<thead>
<tr>
<th>Region</th>
<th>Revenue</th>
<th>Standard Deviation of Revenue</th>
<th>Max Revenue</th>
<th>Min Revenue</th>
<th>Count Distinct (Items Sold)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$2,334,864</td>
<td>$22,000</td>
<td>$3</td>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>$3,413,340</td>
<td>$27,900</td>
<td>$3</td>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>Southeast</td>
<td>$2,016,185</td>
<td>$27,900</td>
<td>$3</td>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>Central</td>
<td>$1,773,270</td>
<td>$30,000</td>
<td>$5</td>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>South</td>
<td>$1,380,991</td>
<td>$20,700</td>
<td>$3</td>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>Northwest</td>
<td>$1,485,182</td>
<td>$34,200</td>
<td>$3</td>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>Southwest</td>
<td>$2,816,334</td>
<td>$32,400</td>
<td>$3</td>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>Web</td>
<td>$1,716,267</td>
<td>$77,400</td>
<td>$3</td>
<td>58</td>
<td>1</td>
</tr>
</tbody>
</table>
This report is saved as **Region**.

The difference between the Dynamic Aggregation - Region report and the Region report is where the metric calculations are performed. In the dynamic aggregation report, the metrics are rolled up in memory, with the data available on the report. In the second report, everything is calculated in the data warehouse at the region level.

MicroStrategy can determine these values easily in both ways. When dynamic aggregation occurs, most values roll up correctly in memory. However, when exceptions occur, incorrect values are not displayed. You can easily recalculate the report to obtain the correct values by removing the attribute from the report entirely.
Subtotals

What are subtotals?

*Totaling* is another function of the report view that you can define. *Subtotals* reflect data rolled up to the selected attribute levels and can be applied dynamically to any report. You can apply subtotals using one of many standard subtotal functions such as, total, count, minimum, maximum, standard deviation, and others. If these simple aggregation functions do not satisfy your particular needs, you can create a customized user-defined subtotal using the Subtotal Editor. For more information, see the *What are user-defined subtotals?* section that follows.

You can apply the subtotal by position, across a level, or using group by.

- Applying a subtotal *across a level* calculates a subtotal across the selected attributes. The subtotal is applied to particular levels-rows, columns, and pages. This really means “group by attributes to the left of the selected attribute.”

In other words, if you have Region and Employee, in that order, on a report (as on the Basic Report), selecting across Employee means group by Region. A subtotal for each Region, totaling the individual Employee-Region values, displays on the report. Likewise, across Region means group by none since there is nothing to the left of it on the report. The result is a grand total. However, if the report is pivoted and the order of the attributes changes, the totals also change. If Employee is pivoted to the left of Region, the across Employee subtotal means group by none.
• The *by position* option means applying the subtotal based on its location on the report. The subtotal is calculated across all attributes and hierarchies on the report. It provides the same behavior as across level, but without selecting a level. Instead, the level is selected dynamically so these subtotals change as you alter the layout of the template. The two choices for by position are All subtotals, meaning “across all attributes,” and Grand Total, translating to “across the leftmost attribute.”

For example, you can choose to subtotal on rows and/or columns. The Basic Report contains the columns Region, Employee, Revenue, Cost, and Profit. You can subtotal by both rows and columns, which provides totals at the employee and region level for each metric.

By default, the by position option is selected.

• *Group by* applies the subtotal by the selected attribute across all other attributes on the template, regardless of position. Group by effectively allows you to use both subtotal and sort by attributes that are not the furthest to the left. The Grand Total check box allows you to also add a subtotal grouped by nothing, effectively calculating a total of all attributes on the template.

If a report contains Region, Category, and Quarter and you group by Region, a Region subtotal always appears, regardless of where Category and Quarter are located with respect to Region. You can also group by multiple attributes. For example, grouping by Region-Category on that report provides a subtotal every time a new Region-Category combination occurs.

Group by works best if the report is sorted by the same attribute used to group the subtotals, regardless of position.
### Subtotals by position example

Open the **Subtotals** report, a sample of which is displayed below. This report is based on the Basic Report, with the addition of the attribute Quarter. Also, a view filter has been added, which includes only quarters 1 and 2 of Year 2002 and the Northeast, Central, and South regions.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$ 60,125</td>
<td>$ 46,020</td>
<td>$14,105</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$ 37,129</td>
<td>$ 28,023</td>
<td>$ 9,106</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$ 30,169</td>
<td>$ 20,990</td>
<td>$ 9,171</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td>$ 38,980</td>
<td>$ 29,328</td>
<td>$ 9,652</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$ 42,125</td>
<td>$ 32,049</td>
<td>$10,076</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$ 39,651</td>
<td>$ 30,058</td>
<td>$ 9,593</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total</strong></td>
<td></td>
<td>$ 256,179</td>
<td>$ 194,476</td>
<td>$61,703</td>
</tr>
<tr>
<td></td>
<td>South</td>
<td>Conner</td>
<td>Beatrice</td>
<td>$ 46,307</td>
<td>$ 34,987</td>
<td>$11,320</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nelson</td>
<td>Arthur</td>
<td>$ 47,746</td>
<td>$ 36,308</td>
<td>$11,438</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pierce</td>
<td>Charles</td>
<td>$ 53,195</td>
<td>$ 40,429</td>
<td>$12,766</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total</strong></td>
<td></td>
<td>$ 147,248</td>
<td>$ 111,724</td>
<td>$35,524</td>
</tr>
<tr>
<td></td>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$ 66,824</td>
<td>$ 50,833</td>
<td>$15,991</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$ 45,174</td>
<td>$ 34,287</td>
<td>$10,887</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$ 76,518</td>
<td>$ 57,508</td>
<td>$19,010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td>$ 51,592</td>
<td>$ 39,012</td>
<td>$12,580</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$ 69,371</td>
<td>$ 52,368</td>
<td>$17,003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$ 62,244</td>
<td>$ 47,195</td>
<td>$15,049</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total</strong></td>
<td></td>
<td>$ 371,723</td>
<td>$ 281,203</td>
<td>$90,520</td>
</tr>
<tr>
<td></td>
<td>South</td>
<td>Conner</td>
<td>Beatrice</td>
<td>$ 52,705</td>
<td>$ 39,964</td>
<td>$12,741</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nelson</td>
<td>Arthur</td>
<td>$ 79,624</td>
<td>$ 59,820</td>
<td>$19,804</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pierce</td>
<td>Charles</td>
<td>$ 76,334</td>
<td>$ 57,755</td>
<td>$18,579</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total</strong></td>
<td></td>
<td>$ 208,663</td>
<td>$ 157,539</td>
<td>$51,124</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total</strong></td>
<td></td>
<td>$ 847,940</td>
<td>$ 641,025</td>
<td>$206,915</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total</strong></td>
<td></td>
<td>$ 1,434,539</td>
<td>$ 1,086,529</td>
<td>$348,010</td>
</tr>
</tbody>
</table>

You can create this report yourself by starting with the **Basic Report**. Note that the Subtotal report has a different format than the Basic Report. The Subtotal report uses the autostyle named SmallType, while Basic Report uses Squares.
Each region is totaled for each quarter, then each quarter is totaled, and finally a grand total is calculated. The subtotals use the by position option. To view how these subtotals are set up, select Subtotals from the Data menu.

You can press F11 to toggle the grand total display for reports in Desktop.

Move Region to the left of Quarter and notice that the subtotals change. Instead of totals by region, by quarter, and then a grand total, the subtotals are calculated by quarter, by region, and then for all attributes (that is, a grand total). This dynamic recalculation is a feature of the subtotal by position option. Return Region to its position between Quarter and Employee.

Subtotals across levels example

Begin with the Subtotals report, and change the by position subtotals to across levels.

To set subtotals across levels

1. Select Data, then Subtotals. The Subtotals dialog box opens.

2. Click Advanced. The Advanced Subtotals dialog box opens.

3. Select Across level. A list of report objects is displayed.

4. Select Region from the list of report objects.

5. Click OK, then OK again to return to the report.
The only totals now are quarterly, as displayed below.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$60,125</td>
<td>$46,020</td>
<td>$14,105</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kelly</td>
<td>$37,129</td>
<td>$28,023</td>
<td>$9,106</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kieferson</td>
<td>$38,169</td>
<td>$28,998</td>
<td>$9,171</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sawyer</td>
<td>$38,980</td>
<td>$29,328</td>
<td>$9,652</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sonder</td>
<td>$42,125</td>
<td>$32,049</td>
<td>$10,076</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yager</td>
<td>$39,651</td>
<td>$30,058</td>
<td>$9,593</td>
</tr>
</tbody>
</table>

South

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conner</td>
<td>Beatrice</td>
<td>$46,307</td>
<td>$34,987</td>
<td>$11,320</td>
<td></td>
</tr>
<tr>
<td>Nelson</td>
<td>Arthur</td>
<td>$47,746</td>
<td>$36,308</td>
<td>$11,438</td>
<td></td>
</tr>
<tr>
<td>Pierce</td>
<td>Charles</td>
<td>$53,195</td>
<td>$40,429</td>
<td>$12,766</td>
<td></td>
</tr>
</tbody>
</table>

Total $586,599 $445,504 $141,095

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2 02</td>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$66,824</td>
<td>$50,833</td>
<td>$15,991</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kelly</td>
<td>$45,174</td>
<td>$34,287</td>
<td>$10,887</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kieferson</td>
<td>$76,518</td>
<td>$57,508</td>
<td>$19,010</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sawyer</td>
<td>$51,592</td>
<td>$39,012</td>
<td>$12,580</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sonder</td>
<td>$69,371</td>
<td>$52,368</td>
<td>$17,003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yager</td>
<td>$62,244</td>
<td>$47,195</td>
<td>$15,049</td>
</tr>
</tbody>
</table>

South

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conner</td>
<td>Beatrice</td>
<td>$52,705</td>
<td>$39,964</td>
<td>$12,741</td>
<td></td>
</tr>
<tr>
<td>Nelson</td>
<td>Arthur</td>
<td>$79,624</td>
<td>$59,820</td>
<td>$19,804</td>
<td></td>
</tr>
<tr>
<td>Pierce</td>
<td>Charles</td>
<td>$76,334</td>
<td>$57,755</td>
<td>$18,579</td>
<td></td>
</tr>
</tbody>
</table>

Total $847,940 $641,025 $206,915

Remember, that across levels means group by attributes to the left of the selected attribute. Since the selected attribute is Region, the only attribute to the left of it is Quarter, hence the quarterly totals.

As you did with the by position example, move Region to the left. Only grand total is displayed, because now there is no attribute to the left of Region.

Return Region to its position between Quarter and Employee.
Subtotals group by example

Begin with the Subtotals report again, which contains subtotals by position. Sort the report by region, by right-clicking Region in the grid and selecting Sort, then Ascending. Notice how the Q1 and Q2 Totals now appear at the bottom of the report.

Move Region to the right, after Employee. The employees for each region are displayed, then employee totals for each quarter, with a quarterly total, and finally a grand total. Now change the by position subtotals to group by.

To set group by subtotals

1. Select Data, then Subtotals. The Subtotals dialog box opens.

2. Click Advanced. The Advanced Subtotals dialog box opens.

3. Select Group by. A blank list of group by levels is displayed.


5. Select Region from the list of attributes on the report.

6. Click OK to return to the Advanced Subtotals dialog box. Notice that Region has been added to the list of levels.

7. Click OK, then OK again to return to the report.
Now the sort and the subtotals work together to provide regional totals, as shown below.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Employee</th>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>Ellerkamp</td>
<td>Nancy</td>
<td>Central</td>
<td>$50,167</td>
<td>$38,099</td>
<td>$12,068</td>
</tr>
<tr>
<td></td>
<td>Gale</td>
<td>Loren</td>
<td>Central</td>
<td>$53,012</td>
<td>$40,333</td>
<td>$12,679</td>
</tr>
<tr>
<td></td>
<td>Torrison</td>
<td>Mary</td>
<td>Central</td>
<td>$32,620</td>
<td>$25,040</td>
<td>$7,580</td>
</tr>
<tr>
<td></td>
<td>Zemlicka</td>
<td>George</td>
<td>Central</td>
<td>$47,373</td>
<td>$35,832</td>
<td>$11,541</td>
</tr>
<tr>
<td>Q2 02</td>
<td>Ellerkamp</td>
<td>Nancy</td>
<td>Central</td>
<td>$50,307</td>
<td>$37,932</td>
<td>$12,375</td>
</tr>
<tr>
<td></td>
<td>Gale</td>
<td>Loren</td>
<td>Central</td>
<td>$106,610</td>
<td>$80,466</td>
<td>$26,144</td>
</tr>
<tr>
<td></td>
<td>Torrison</td>
<td>Mary</td>
<td>Central</td>
<td>$48,515</td>
<td>$36,750</td>
<td>$11,765</td>
</tr>
<tr>
<td></td>
<td>Zemlicka</td>
<td>George</td>
<td>Central</td>
<td>$62,122</td>
<td>$47,135</td>
<td>$14,987</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>Central</td>
<td>$450,726</td>
<td>$341,587</td>
<td>$109,139</td>
</tr>
<tr>
<td>Q1 02</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>Northeast</td>
<td>$60,125</td>
<td>$46,020</td>
<td>$14,105</td>
</tr>
<tr>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>Northeast</td>
<td>$62,244</td>
<td>$47,195</td>
<td>$15,049</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>Northeast</td>
<td>$627,902</td>
<td>$475,679</td>
<td>$152,223</td>
</tr>
<tr>
<td>Q1 02</td>
<td>Conner</td>
<td>Beatrice</td>
<td>South</td>
<td>$46,307</td>
<td>$34,987</td>
<td>$11,320</td>
</tr>
<tr>
<td></td>
<td>Nelson</td>
<td>Arthur</td>
<td>South</td>
<td>$47,746</td>
<td>$36,308</td>
<td>$11,438</td>
</tr>
<tr>
<td></td>
<td>Pierce</td>
<td>Charles</td>
<td>South</td>
<td>$53,195</td>
<td>$40,429</td>
<td>$12,766</td>
</tr>
<tr>
<td>Q2 02</td>
<td>Conner</td>
<td>Beatrice</td>
<td>South</td>
<td>$52,705</td>
<td>$39,964</td>
<td>$12,741</td>
</tr>
<tr>
<td></td>
<td>Nelson</td>
<td>Arthur</td>
<td>South</td>
<td>$79,624</td>
<td>$59,820</td>
<td>$19,804</td>
</tr>
<tr>
<td></td>
<td>Pierce</td>
<td>Charles</td>
<td>South</td>
<td>$76,334</td>
<td>$57,755</td>
<td>$18,579</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>South</td>
<td>$355,911</td>
<td>$269,263</td>
<td>$86,648</td>
</tr>
</tbody>
</table>

What are custom subtotals?

By default, when you use subtotals in a report, the same subtotal function is used for all metrics in the report. The name of the subtotal is displayed in the subtotal line items that appear in the report. You can use custom subtotals to give you more control over the characteristics of a subtotal. Custom subtotals allow you to define custom subtotal line items that appear on your reports. Custom subtotals allow you to do the following:

- customize the subtotal name that appears in the subtotal line item.
- define different subtotal functions to be used on different metrics in the report.
specify the level of each total.
• turn off subtotaling for specific metrics on the report.

You can make the subtotal name dynamic by typing special characters in the subtotal name field as listed in the following table.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#A</td>
<td>The name of the attribute under which the subtotal appears.</td>
</tr>
<tr>
<td>#P</td>
<td>The name of the attribute to the left of, or above the attribute under which the subtotal appears.</td>
</tr>
<tr>
<td>#0</td>
<td>All the forms of the parent element.</td>
</tr>
<tr>
<td>#1</td>
<td>The first form of the parent element reading from left to right or from top to bottom.</td>
</tr>
<tr>
<td>#2</td>
<td>The second form of the parent element reading from left to right or from top to bottom.</td>
</tr>
<tr>
<td>#3</td>
<td>The third form of the parent element reading from left to right or from top to bottom.</td>
</tr>
<tr>
<td>#4</td>
<td>The fourth form of the parent element reading from left to right or from top to bottom.</td>
</tr>
</tbody>
</table>

An attribute form provides details that identify and describe an attribute. Examples are an ID, a description, or a name. For more information, see Chapter 11, Attributes.

**Custom subtotal example**

Open the Subtotals report from the previous example. You can add custom subtotals for the region and quarter by following the steps outlined below.
To add custom subtotals

1. Select **Subtotals** from the **Data** menu. The Subtotals dialog box opens.

2. Clear the **Totals** check box to remove the standard subtotals.

3. Click **Advanced**, then **New** to create a custom subtotal.

4. Type the following for the name:
   
   **Total for the #P #0**

   Remember that P displays the parent attribute and 0 (the number zero, not the letter o) displays all the forms of the parent attribute. In this case, only one form exists for each.

5. All the metrics on the report are listed. You can select the subtotal function to use for each. Total is correct for all of our metrics.

6. Click **OK** to save the new subtotal.

7. Click **OK** to return to the Subtotals dialog box.

8. Select the **Total for the #P #0** check box. Notice the icon for this custom subtotal is different from those of the prebuilt subtotals.

9. Click **Advanced**. On the Advanced Subtotals Options dialog box, select **Across level**, and then select the check boxes **Region**, and **Employee**.

10. Create another custom subtotal, called Grand Total. Do not change the subtotal functions for any of the metrics.

11. Select the **Grand Total** check box.

12. Select **Across level** and **Quarter**.

13. Click **OK** to return to the Subtotals dialog box.

14. Click **OK**.
The report results are displayed below.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$60,125</td>
<td>$46,020</td>
<td>$14,105</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$37,129</td>
<td>$28,023</td>
<td>$9,106</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$38,169</td>
<td>$28,998</td>
<td>$9,171</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td>$38,980</td>
<td>$29,328</td>
<td>$9,652</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$42,125</td>
<td>$32,049</td>
<td>$10,076</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$39,651</td>
<td>$30,058</td>
<td>$9,593</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for the Region Northeast</strong></td>
<td>$256,179</td>
<td>$194,476</td>
</tr>
<tr>
<td></td>
<td>South</td>
<td>Conner</td>
<td>Beatrice</td>
<td>$46,307</td>
<td>$34,987</td>
<td>$11,320</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nelson</td>
<td>Arthur</td>
<td>$47,746</td>
<td>$36,308</td>
<td>$11,438</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pierce</td>
<td>Charles</td>
<td>$53,195</td>
<td>$40,429</td>
<td>$12,766</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for the Region South</strong></td>
<td>$147,248</td>
<td>$111,724</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for the Quarter Q1 02</strong></td>
<td>$586,599</td>
<td>$445,504</td>
</tr>
<tr>
<td>Q2 02</td>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$66,824</td>
<td>$50,833</td>
<td>$15,991</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$45,174</td>
<td>$34,287</td>
<td>$10,887</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pierce</td>
<td>Charles</td>
<td>$76,334</td>
<td>$57,755</td>
<td>$18,579</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for the Region South</strong></td>
<td>$208,663</td>
<td>$157,539</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for the Quarter Q2 02</strong></td>
<td>$847,940</td>
<td>$641,025</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Grand Total</td>
<td></td>
<td><strong>Total</strong></td>
<td>$1,434,539</td>
<td>$1,086,529</td>
</tr>
</tbody>
</table>

This report is saved as **Custom Subtotals**.

**What are user-defined subtotals?**

The standard predefined subtotal functions, which are automatically available for use with every metric and report, are simple aggregate functions that satisfy many subtotaling requirements. If they do not answer your specific needs, you can create a **user-defined subtotal** using the Subtotal Editor. User-defined subtotals allow you to define new subtotal functions. You can then use these function expressions in subtotal definitions just like any of the built-in subtotal functions (for example, Total, Count, Average).
You can create your own subtotal using any combination of the following:

- an aggregation function, such as avgdev, IRR, MIRR, and NPV, that is not one of the standard predefined subtotal functions
- multiple functions
- constants in conjunction with aggregation functions
- nested functions
- dimensional subtotals
- other metrics in the subtotal formula

For example, you need a subtotal that always calculates at the year level, regardless of the level of the report. You can create a user-defined subtotal, setting it to the Year level (or dimension). Another example is using a weighted subtotal, where a subtotal is weighted with another metric, such as a average profit weighted by units sold. This example is included in the User-defined subtotal example (weighted subtotal) below.

After it is created and applied to a metric, a user-defined subtotal is indistinguishable from the standard predefined subtotal functions such as total or count. For example, it can be used in a metric as the total subtotal function, which calculates the metric's totals when the metric is used on a report. The dynamic aggregation function, which is used when the Analytical Engine aggregates the metric, can also be set to a user-defined subtotal.

Do not confuse user-defined subtotals with custom subtotals, which are defined for a particular report for display purposes. User-defined subtotals are made available to metrics and can then be used on any report that uses the metric.
User-defined subtotal example (First function)

You need to produce an inventory report showing the number of electronics products received in 2003, by month and by product. The report must also provide the first shipment amount. To do this, create a user-defined subtotal using the First function. Create a metric with the Units Received fact and set the Total subtotal function, which calculates the metric’s grand totals, to the new subtotal. Finally, create the report. When you apply subtotals, the user-defined subtotal displays the amount of items received in the first shipment, regardless of the month in which it arrived.

Since this example focuses on using the Subtotal Editor and user-defined subtotals, a detailed procedure is provided to create the subtotal. The instructions to create the metric and report are less comprehensive. For details, refer to the online help.

To create and use a user-defined subtotal

1. On the Desktop, from the File menu, point to New, and then select Subtotal. The Subtotal Editor opens. Notice its resemblance to the Metric Editor—theyir similarity in function and appearance helps to ease the process of creating subtotals.

2. In the Object Browser, navigate to the Basic Functions folder, found under Functions and Operators. Select the First function.

3. In the formula definition, type \( x \), which is the placeholder for the metric to be subtotaled.

4. Right-click First in the definition and select First parameters. The First Parameters dialog box opens.

5. The First function must be sorted to achieve correct results. Click the Sort By tab.

6. Select Sort by objects, then click Add. The Select Objects dialog box opens.
7 Since our report is sorted by time, double-click the following, in order, to add them to the sort:

- Year
- Quarter
- Month
- Day

8 Click OK, then OK again to return to the Subtotal Editor.

For more information on the First function, including details on sorting, see the MicroStrategy Analytical Engine Functions Reference.

9 Click Validate. You should receive a “Valid expression” message in the status area.

10 Click Save and Close. Name the new subtotal First (Date Sort). You are returned to the Desktop.

11 Create a new metric:

- In the Metric Editor, select the Units Received fact.
- On the Subtotals/Aggregation tab, select First (Date Sort) as the Total subtotal function.
- Save the metric as Units Received.

12 Create a new report:

- In the Report Editor, place Item on the rows, and then Month and the Units Received metric on the columns.
- Filter on Year = 2003 and Category = Electronics.
- Add Grand totals.
- Execute the report. The results are displayed in the following report sample:
This sample presents only a subset of the entire report, showing the first three months of the year and the total.

Notice that the Total does not add all units received throughout the year, but rather displays the amount of the first shipment, regardless of what month the shipment arrived. For example, you received 20 AM/FM Stereo Receivers each month; the total is 20, the amount received in January alone. No Digital Surround Sound Receivers were received in January, but 20 were received in February, which the subtotal reflects. Hitachi Hi8 Camcorders, in the last line of the sample, were not received until March, so the total is derived from the March shipment.

Remember to save the report if you want to keep it.

<table>
<thead>
<tr>
<th>Item</th>
<th>Metrics Month</th>
<th>Units Received Jan 03</th>
<th>Units Received Feb 03</th>
<th>Units Received Mar 03</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harman Kardon Digital Surround Sound Receiver</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Harman Kardon AM/FM Stereo Receiver</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Harman Kardon Dolby Digital Receiver</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>GPX Portable CD Player with Bass Boost</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>GPX Portable CD Player with Car Kit</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>GPX CD AM/FM Cassette Recorder Karaoke Machine</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Panasonic Portable CD Player</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Pioneer 25-Disc Changer with Remote Control</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Pioneer CD Recordable, Digital Synchro Recording</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Pioneer Dolby Pro Logic Receiver</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>RCA CD Changer</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Sony Discman</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Sony Boombox with Digital Tuner</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Sony Digital MiniDisc</td>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Hitachi Hi8 Camcorder</td>
<td></td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>
**User-defined subtotal example (weighted subtotal)**

You need to create a report containing units sold and profit by quarter. For each year, the report must calculate the average units sold and the average profit, weighted by the number of units sold during the year. Since these two averages must display on the same line, a custom subtotal is needed. The formula for the weighted yearly average profit is:

\[
\text{Sum}(x \times [\text{Units Sold}])\{\text{Year}\}/\text{Sum}([\text{Units Sold}])\{\text{Year}\}
\]

where

- \( x \) is the placeholder for the metric to be subtotalled. In this case, it will be the Profit metric.
- \([\text{Units Sold}]\) is a metric that sums the fact Units Sold.
- \( \{\text{Year}\} \) is the level of the subtotal calculation.

Finally, you need to sum the weighted yearly average profit over all the years. The formula for this subtotal is:

\[
\text{Sum}(\text{Sum}(x \times [\text{Units Sold}])\{\text{Year}\}/\text{Sum}([\text{Units Sold}])\{\text{Year}\})\{}
\]

which calculates the sum on an empty level, represented by the \( \{} \). An empty level calculates across the entire report. This subtotal will be used as a grand total on the report. You need to create user-defined subtotals for both of these formulas because they are not standard subtotal functions.

As with the previous example, the focus is using the Subtotal Editor and user-defined subtotals, so a detailed procedure is provided to create the subtotal. The instructions to create the metric and report are less comprehensive. For details, refer to the online help.
To create and use a user-defined subtotal

1. On the Desktop, select File, point to New, and then Subtotal. The Subtotal Editor opens.

2. Type the formula for the weighted yearly average subtotal, as displayed and described above.

3. Click Validate. In the Ambiguity: ‘Units Sold’ dialog box, select the metric and click OK. You should then receive a “Valid expression” message in the status area.

4. Click Save and Close. Name the new subtotal Weighted Yearly Average. You are returned to the Desktop.

5. Open the Subtotal Editor again.

6. Type the formula for the sum of the weighted yearly average subtotal, as displayed and described above.

7. Click Validate. In the Ambiguity: ‘Units Sold’ dialog box, select the metric and click OK. You should then receive a “Valid expression” message in the status area.

8. Click Save and Close. Name the new subtotal Sum of WYA (which stands for Weighted Yearly Average). You are returned to the Desktop.

9. Open the Profit metric in the Metric Editor.

10. Click the Subtotals/Aggregation tab.

11. Select Weighted Yearly Average and Sum of WYA in the Available project subtotals list. Click > to add them to the metric.

12. Save the metric and close the Metric Editor.

13. Create the new report:

   – In the Report Editor, place Year and Quarter on the rows, and then the Units Sold and Profit metrics on the columns.
Now we are going to create the subtotals for the report. Select Subtotals from the Data menu.

Click Advanced.

Click New to create a new custom subtotal, which will contain a standard average for Units Sold and the weighted yearly average subtotal for Profit.

Type Average, Weighted Yearly Average as the name of the custom subtotal.

For Units Sold, select Average from the pull-down list.

For Profit, select Weighted Yearly Average.

Click OK.

Select Across level and then Quarter, so the subtotals are calculated for each year.

Next, create another custom subtotal to display in the grand total position. Click New.

Type Overall Average, Sum of WYA as the name.

For Units Sold, select Average from the pull-down list.

For Profit, select Sum of WYA.

Click OK.

Select By position. For Rows, select Grand Total from the pull-down list. For both Columns and Pages, select None.

Click OK. Notice your two custom subtotals are selected.

Notice that custom subtotals are distinguished by a different icon. An icon with an exclamation mark (!) means that the subtotals are not available for all metrics on the report. Recall that you added the user-defined subtotals to the Profit metric only, not the Units Sold metric.

Click OK to return to the Report Editor.
What are smart subtotals?

A compound metric, at a high-level, is composed of two metrics, such as Metric1/Metric2. The subtotal of a compound metric can be calculated in two different ways:

- Calculate the sum of all parts of the compound metric, then perform the compound metric. This formula is represented by \( \frac{\text{Sum(Metric1)}}{\text{Sum(Metric2)}} \).

- Calculate the compound metric for each row of the report, and then roll up the data to the correct level. The formula for this is \( \text{Sum(Metric1/Metric2)} \).

The first case uses *smart subtotals*, which calculate subtotals on the individual elements of a metric. For example, the Profit Margin metric is calculated as the Profit metric divided by the Revenue metric. The Profit Margin metric can be totaled as follows:

- Add all the profit values together. Add all the revenue values together. Divide the two sums. This is a smart metric.

- Divide each profit value by each revenue value. Sum up these ratios.
The sample report clearly illustrates the difference between these two methods of totaling.

Smart subtotals are also referred to as smart metrics. The smart metric setting is applied in the Metric Editor. For more information, see Chapter 6, *Metrics*.

**Smart subtotal example**

Edit the **Custom Subtotals** report. Since the **Cost** metric is not a part of the profit margin calculations, move the **Cost** metric to Report Objects so that it is no longer displayed on the grid. Open the **Supporting Objects** folder. Add the **Profit Margin** metric to the grid. Add the **Profit Margin (Smart)** metric to the grid. Remove the custom subtotals (Total for #P #0 and Grand Total) added previously. Select **Total** and execute the report. The results are displayed as shown in the following figure.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Region</th>
<th>Employee</th>
<th>Revenue</th>
<th>Profit</th>
<th>Profit Margin</th>
<th>Profit Margin (Smart)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>Northeast</td>
<td>De Le Torre Sandra</td>
<td>$60,125</td>
<td>$14,105</td>
<td>23.46%</td>
<td>23.46%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kelly Laura</td>
<td>$37,129</td>
<td>$9,106</td>
<td>24.53%</td>
<td>24.53%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kieferon Jack</td>
<td>$38,169</td>
<td>$9,171</td>
<td>24.03%</td>
<td>24.03%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sawyer Leanne</td>
<td>$38,980</td>
<td>$9,652</td>
<td>24.76%</td>
<td>24.76%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sonder Melanie</td>
<td>$42,125</td>
<td>$10,076</td>
<td>23.92%</td>
<td>23.92%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yager Beth</td>
<td>$39,651</td>
<td>$9,593</td>
<td>24.19%</td>
<td>24.19%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>$256,179</td>
<td>$61,703</td>
<td>24.09%</td>
<td>24.09%</td>
</tr>
<tr>
<td></td>
<td>Central</td>
<td>Ellerkamp Nancy</td>
<td>$50,167</td>
<td>$12,068</td>
<td>24.06%</td>
<td>24.06%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gale Loren</td>
<td>$53,012</td>
<td>$12,679</td>
<td>23.92%</td>
<td>23.92%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Torrison Mary</td>
<td>$32,620</td>
<td>$7,580</td>
<td>23.24%</td>
<td>23.24%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zemlicka George</td>
<td>$47,373</td>
<td>$11,541</td>
<td>24.36%</td>
<td>24.36%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>$183,172</td>
<td>$43,668</td>
<td>23.95%</td>
<td>23.95%</td>
</tr>
<tr>
<td></td>
<td>South</td>
<td>Conner Beatrice</td>
<td>$46,307</td>
<td>$11,320</td>
<td>24.45%</td>
<td>24.45%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nelson Arthur</td>
<td>$47,746</td>
<td>$11,438</td>
<td>23.96%</td>
<td>23.96%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pierce Charles</td>
<td>$53,195</td>
<td>$12,766</td>
<td>24.00%</td>
<td>24.00%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>$147,248</td>
<td>$35,524</td>
<td>24.13%</td>
<td>24.13%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>$586,599</td>
<td>$141,095</td>
<td>24.05%</td>
<td>24.05%</td>
</tr>
</tbody>
</table>
Both profit margin metrics provide the same value for De Le Torre ($14,105/$60,125 or 23.46%). However, look at the first total, for Q1 2002/Northeast. Total profit is $61,703, and total revenue is $256,179. If you calculate the profit margin using smart totals, the formula is $61,703/$256,179, or 24.09%. The alternative adds the profit margin values for the quarter and region, arriving at 144.89%, which is, of course, not a valid percentage.

**Shortcut metrics**

**What are shortcut metrics?**

*Shortcut metrics* are based on metrics already included in a report and provide a quick way to add new metrics to a report. They are really just special derived metrics. Shortcut metrics are available when you right-click on a metric column or metric header and are based on the selected metric. Shortcut metrics can be found in the Desktop only.

Shortcut metrics belong to one of the following categories:

- Percent-to-total metrics: display a percent in relation to a selected total of each item affected by the metric.
- Transformation metrics: apply offset values, such as “four months ago,” to the selected attribute.
- Rank metrics: apply a ranking number to the metric values for a given attribute.

For details and examples of shortcut metrics, refer to Chapter 6, *Metrics*. 
Advanced sorting

Sorting allows you to order the report data set to present your business information in a more informative way. For example, you can alphabetically sort country and region on a report, allowing you to quickly find a particular region. The Basic Reporting Guide discusses such quick sorting, which is selecting a column or row to sort on.

Advanced sorting allows you to create your own, more complex sorts for rows and columns. You can select the object to sort by, the sorting order (ascending or descending), the sorting criteria, and the position of the totals. The options for the sorting criteria depend on the sort object. For example, Employee can be sorted by last name, first name, Social Security Number, or the attribute ID. The sorting criteria do not have to be displayed on the report.

Multiple-key sorting, or hierarchical sorting, allows you to sort data according to multiple sorting criteria in a hierarchical manner. This means that the first criterion is the basis for sorting. Any ties are resolved using the second criterion, any remaining ties are resolved using the third criterion, and so on. If a tie remains after all the criteria are used, the default sort order is used as the tiebreaker. In a simple example, you can sort by ascending employee last name, then ascending employee first name. If two employees have the same last name, their first names are compared to alphabetically sort them. You can, of course, create more complex multiple-key sorting.

Sorting metrics hierarchically allows you to use group totals for sorting. That is, the groups on the report are totaled, and the report is sorted on these totals. An example of hierarchical sorting is explained after the advanced sorting example that follows.
**Advanced sorting example**

Open the **Advanced Sorting** report, a subset of which is shown below. While you can create this report yourself, there are many features on it, so it is quicker to just examine the completed report.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Quarter Metrics</th>
<th>Q4 03 Revenue</th>
<th>Q4 03 Rank</th>
<th>Q3 03 Revenue</th>
<th>Q3 03 Rank</th>
<th>Q2 03 Revenue</th>
<th>Q2 03 Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central</td>
<td>Gale</td>
<td>Loren</td>
<td>$ 118,093</td>
<td>4</td>
<td>$ 36,165</td>
<td>2</td>
<td>$ 73,570</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Zemlicka</td>
<td>George</td>
<td>$ 74,852</td>
<td>3</td>
<td>$ 40,801</td>
<td>3</td>
<td>$ 62,429</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Ellerkamp</td>
<td>Nancy</td>
<td>$ 52,581</td>
<td>2</td>
<td>$ 54,331</td>
<td>4</td>
<td>$ 48,054</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Torrison</td>
<td>Mary</td>
<td>$ 55,606</td>
<td>1</td>
<td>$ 35,643</td>
<td>1</td>
<td>$ 46,788</td>
<td>1</td>
</tr>
<tr>
<td>Northwest</td>
<td>Becker</td>
<td>Kyle</td>
<td>$ 108,545</td>
<td>3</td>
<td>$ 67,766</td>
<td>3</td>
<td>$ 94,331</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Gedot</td>
<td>Harriet</td>
<td>$ 74,870</td>
<td>2</td>
<td>$ 37,496</td>
<td>2</td>
<td>$ 59,647</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Hall</td>
<td>David</td>
<td>$ 47,759</td>
<td>1</td>
<td>$ 26,625</td>
<td>1</td>
<td>$ 49,579</td>
<td>1</td>
</tr>
</tbody>
</table>

Click **Advanced Sorting** under the Data menu. The rows are sorted by ascending region and descending fourth quarter 2003 revenue. The columns are sorted by the quarter ID in descending order. Return to the report and examine the sorted data. Notice, that the columns are in reverse order, fourth quarter 2003 to first quarter 2002. The customized banding makes it easier to view the region separations in the rows. Notice that the regions are in alphabetical order, Central to Web. The rank metric helps you confirm that within each region, employees are sorted based on fourth quarter 2003 revenue. For example, the rank is 4, 3, 2, 1 in the Central region for Q4 03. For Q3 03, the rank is 2, 3, 4, 1.

**Hierarchical sorting example**

On the **Advanced Sorting** report used in the previous example, complete the following steps to prepare for the hierarchical sorting example. These tasks are not needed to sort a report hierarchically, only on these sample reports.


2. Move **Quarter** from the columns to the rows, to the left of Region.
3  Edit the view filter to remove Northwest and Web from the list of regions.

4  Add standard totals by choosing Subtotals from the Data menu, then selecting Totals from the list of available subtotals.

The following procedure sorts the report by revenue, in descending order. The totals are placed at the top of each section, rather than more conventionally at the bottom.

To sort metrics hierarchically

1  Select Advanced Sorting from the Data menu. The Sorting dialog box opens.

2  On the Rows tab, click Remove All to delete the previous sort.

3  Click Add to create a new sort.

4  Change Sort By to Revenue.

5  Change the Order to Descending.

6  Change the Total Position to Top.

7  Select Sort metrics hierarchically and choose Total.

8  Click OK.
The results are displayed below.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>$5,170,447</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td></td>
<td>$932,383</td>
</tr>
<tr>
<td></td>
<td>McClain</td>
<td>Sean</td>
<td></td>
<td>$115,108</td>
</tr>
<tr>
<td></td>
<td>Strome</td>
<td>Fred</td>
<td></td>
<td>$108,615</td>
</tr>
<tr>
<td></td>
<td>Benner</td>
<td>Ian</td>
<td></td>
<td>$80,248</td>
</tr>
<tr>
<td></td>
<td>Lynch</td>
<td>Sam</td>
<td></td>
<td>$72,490</td>
</tr>
<tr>
<td>Q4 02</td>
<td>Southeast</td>
<td></td>
<td></td>
<td>$317,558</td>
</tr>
<tr>
<td></td>
<td>Gale</td>
<td>Loren</td>
<td></td>
<td>$111,641</td>
</tr>
<tr>
<td></td>
<td>Zemlicka</td>
<td>George</td>
<td></td>
<td>$77,922</td>
</tr>
<tr>
<td></td>
<td>Ellerkamp</td>
<td>Nancy</td>
<td></td>
<td>$77,680</td>
</tr>
<tr>
<td></td>
<td>Torrison</td>
<td>Mary</td>
<td></td>
<td>$50,315</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td></td>
<td>$236,364</td>
</tr>
<tr>
<td></td>
<td>Pierce</td>
<td>Charles</td>
<td></td>
<td>$97,826</td>
</tr>
<tr>
<td></td>
<td>Nelson</td>
<td>Arthur</td>
<td></td>
<td>$80,650</td>
</tr>
<tr>
<td></td>
<td>Conner</td>
<td>Beatrice</td>
<td></td>
<td>$59,888</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td></td>
<td>$927,290</td>
</tr>
<tr>
<td>Q4 03</td>
<td>Southeast</td>
<td></td>
<td></td>
<td>$387,515</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td></td>
<td>$311,132</td>
</tr>
<tr>
<td></td>
<td>Strome</td>
<td>Fred</td>
<td></td>
<td>$111,800</td>
</tr>
<tr>
<td></td>
<td>McClain</td>
<td>Sean</td>
<td></td>
<td>$103,076</td>
</tr>
<tr>
<td></td>
<td>Benner</td>
<td>Ian</td>
<td></td>
<td>$88,616</td>
</tr>
<tr>
<td></td>
<td>Lynch</td>
<td>Sam</td>
<td></td>
<td>$84,023</td>
</tr>
<tr>
<td></td>
<td>Central</td>
<td></td>
<td></td>
<td>$118,093</td>
</tr>
<tr>
<td></td>
<td>Gale</td>
<td>Loren</td>
<td></td>
<td>$74,852</td>
</tr>
<tr>
<td></td>
<td>Zemlicka</td>
<td>George</td>
<td></td>
<td>$62,581</td>
</tr>
<tr>
<td></td>
<td>Ellerkamp</td>
<td>Nancy</td>
<td></td>
<td>$55,606</td>
</tr>
</tbody>
</table>

This report is saved as Advanced Sorting - Hierarchical.

Notice how the report is sorted. Within the region Southeast in Q4 2002, the employees are sorted by revenue, in the order of the highest revenue producer to the lowest. Within Q4 2002, the regions are also sorted, from Southeast with $376,461 in revenue to South with only $238,364. The quarters are sorted, from Q4 2002 at $932,383 to Q1 2003 at $121,639. The groups on the report are sorted hierarchically.
Formatting

You can change the general presentation formats and formatting details of a report to suit your requirements and preferences. The Formatting toolbar allows you to set various formatting properties for row and column headers, as well as for the actual report data. You also can set borders and patterns. For more information on formatting basics, see the Reporting Essentials chapter of the Basic Reporting Guide. You can also set the formatting options for a report through the grid view or design view using the Format Cells dialog box. For details on how to access the Format Cells dialog box, see the online help.

Formatting Cells dialog box

The Format Cells dialog box consists of the following tabs:

**Number**- Allows you to select the number formatting options, such as decimal spaces, currency symbol, time format, zip code format, and so on. If none of the built-in number formats meet your needs, you can create your own custom number formats using number format symbols. For more details on custom formatting see “Custom Formats” starting on page 79.

**Alignment**- Determines how the contents of the section are aligned when the formatting is applied. You can select horizontal and vertical alignment, as well as select if you would like to wrap the text or not.

**Font**- Allows you to define the text font for the selected section. You can select the font name, font style, size, color, and effects.

**Border**- Allows you to define how the borders are displayed for the selected section.

**Pattern**- The pattern settings define how to fill the cell background.
Custom Formats

Custom formats allow you to create your own formats for data in a report. You can format text, numbers, and date and time using custom formats. Once you create a custom format, you can use it in other metrics and report objects as well. Each custom format can have up to four optional sections, one each for:

- positive numbers
- negative numbers
- zero values
- text

You can specify these sections, separated by semicolons in the order listed above. If you specify only two sections, the first is used for positive numbers and zeros, and the second is used for negative numbers. If you specify only one section, all numbers use the same format. The following paragraphs list the different custom formats that you can apply for text, numeric data, and date and time with examples of each type of formatting.
## Numeric Data

You can format fractions or numbers with decimal points by including appropriate digit placeholders in the custom format. This is explained in detail in the following table:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0(zero) | Digit placeholder.  
  • If the number contains fewer digits than the placeholders contained in the format, the number is padded with zeros.  
  For example, the format code 00000 will display the number 12 as 00012.  
  • If there are more digits to the right of the decimal point than the placeholders in the format, the decimal portion is rounded to the number of places specified by the placeholders.  
  • If there are more digits to the left of the decimal point than the placeholders in the format, the extra digits are retained.  
  • If the format contains zeros to the left of the decimal point, numbers less than one are displayed with a zero to the left of the decimal point. |
| #      | Digit placeholder.  
  • This digit placeholder displays only significant digits and does not display insignificant zeros.  
  For example, the format code ##.## will display the number 0025.630 as 25.63.  
  • If there are more digits to the right of the decimal point than the placeholders in the format, the decimal portion is rounded to the number of places specified by the placeholders.  
  • If there are more digits to the left of the decimal point than the placeholders in the format, the extra digits are retained.  
  • If the format contains only number signs (#) to the left of the decimal point, numbers less than one are displayed beginning with a decimal point.  
  The format #.00 will display the number 0.43 as .43. |
| ?      | Digit placeholder.  
  • This digit placeholder adds spaces for insignificant zeros on either side of the decimal point so that decimal points align when formatted with a fixed-width font.  
  • You can also use ? for fractions that have varying numbers of digits. |
Character/text data

You can include formats for text and character data as mentioned in the following table:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>This symbol displays the number as a percentage, by multiplying the number by 100 and appending the % character.</td>
</tr>
</tbody>
</table>
| , (comma) | Thousands separator.  
- If the format contains commas separated by #’s or 0’s, commas separate the thousands.  
- A comma following a placeholder scales the number by a thousand. For example, using 0,, scales the number by 1000, so that 10,000 displays as 10. |
| E+, E-, e+, e- | Scientific notation.  
- If the format contains a scientific notation symbol to the left of a 0 or # placeholder, the number is displayed in scientific notation and an E or e is added.  
- The number of 0 and # placeholders to the right of the decimal determines the number of digits in the exponent.  
- E- and e- place a minus sign by negative exponents. E+ and e+ place a minus sign by negative exponents and a plus sign by positive exponents. |
| $ - + / ( ) : space, !, & , ~ , { }, =, < >, ^ | These characters are displayed without the use of quotation marks.  
To display a character other than those listed, precede the character with a back slash (\) or enclose it in double quotation marks (" "). You can also use the slash (/) for fraction formats. |
| *(asterisk) | This symbol repeats the next character until the width of the column is filled. Only one asterisk can be used in each format section. |
| _(underline) | This symbol skips the width of the next character. For example, to make negative numbers surrounded by parentheses align with positive numbers, you can include the format _) for positive numbers to skip the width of a parenthesis. |
Date and Time

The format codes for formatting days, months, years and time in a report are given in the following table:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;text&quot;</td>
<td>Displays the text inside the quotation marks.</td>
</tr>
<tr>
<td>@</td>
<td>Text placeholder. Any text in the cell replaces the @ format symbol. For example, if you want the headers of the Revenue and Profit columns in a report to display as &quot;Revenue This Month&quot;, and &quot;Profit This Month&quot;, type the format code as @ &quot; This Month&quot; and apply it to the Revenue metric and Profit metric.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>Month number. Displays the month as digits without leading zeros, such as 1. Can also represent minutes when used with h or hh formats.</td>
</tr>
<tr>
<td>mm</td>
<td>Month number. Displays the month as digits with leading zeros, as in 01. Can also represent minutes when used with the h or hh formats.</td>
</tr>
<tr>
<td>mmm</td>
<td>Month abbreviation, such as Jan.</td>
</tr>
<tr>
<td>mmmm</td>
<td>Month name, such as January.</td>
</tr>
<tr>
<td>d</td>
<td>Day number. Displays the day as digits with no leading zero, such as 1.</td>
</tr>
<tr>
<td>dd</td>
<td>Day number. Displays the day as digits with leading zeros, as in 01.</td>
</tr>
<tr>
<td>ddd</td>
<td>Day abbreviation, such as Sun.</td>
</tr>
<tr>
<td>dddd</td>
<td>Day name, such as Sunday.</td>
</tr>
<tr>
<td>yy</td>
<td>Year number. Displays the year as a two-digit number, such as 00.</td>
</tr>
<tr>
<td>yyyy</td>
<td>Year number. Displays the year as a four-digit number, such as 2003.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>g</td>
<td>If you are using a Japanese locale, displays the Latin letter for an era.</td>
</tr>
<tr>
<td>gg</td>
<td>If you are using a Japanese locale, displays the first character of an era name.</td>
</tr>
<tr>
<td>ggg</td>
<td>If you are using a Japanese locale, displays the full era name.</td>
</tr>
<tr>
<td>e</td>
<td>If you are using a Japanese locale, displays the full era year.</td>
</tr>
<tr>
<td>ee</td>
<td>If you are using a Japanese locale, displays the full era year with a leading zero if the year is less than ten.</td>
</tr>
<tr>
<td>h</td>
<td>Hour number. Displays the hour as a number without leading zeros, such as 1. If the format contains an AM or PM format, the hour is based on a 12-hour clock; otherwise, it is based on a 24-hour clock.</td>
</tr>
<tr>
<td>hh</td>
<td>Hour number. Displays the hour as a number with leading zeros, as in 01. If the format contains an AM or PM format, the hour is based on a 12-hour clock; otherwise, it is based on a 24-hour clock.</td>
</tr>
<tr>
<td>m</td>
<td>Minute number. Displays the minute as a number without leading zeros, such as 0. The m format must appear immediately after the h or hh symbol; otherwise it is interpreted as month.</td>
</tr>
<tr>
<td>mm</td>
<td>Minute number. Displays the minute as a number with leading zeros, such as 00. The mm format must appear immediately after the h or hh symbol; otherwise it is interpreted as month.</td>
</tr>
<tr>
<td>s</td>
<td>Second number. Displays the second as a number without leading zeros, such as 0.</td>
</tr>
<tr>
<td>ss</td>
<td>Second number. Displays the second as a number with leading zeros, such as 00.</td>
</tr>
<tr>
<td>AM/PM</td>
<td>12-hour time. Displays time using a 12-hour clock. Displays AM, am, A, or “a” to display time between midnight and noon; displays PM, pm, P, or p to display time between noon and midnight.</td>
</tr>
</tbody>
</table>
You can change the color of data in your report using custom formatting. The following table lists the format for color codes:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Black]</td>
<td>Displays cell text in black.</td>
</tr>
<tr>
<td>[Blue]</td>
<td>Displays cell text in blue.</td>
</tr>
<tr>
<td>[Cyan]</td>
<td>Displays cell text in cyan.</td>
</tr>
<tr>
<td>[Green]</td>
<td>Displays cell text in green.</td>
</tr>
<tr>
<td>[Magenta]</td>
<td>Displays cell text in magenta.</td>
</tr>
<tr>
<td>[Red]</td>
<td>Displays cell text in red.</td>
</tr>
<tr>
<td>[White]</td>
<td>Displays cell text in white.</td>
</tr>
<tr>
<td>[Yellow]</td>
<td>Displays cell text in yellow.</td>
</tr>
<tr>
<td>[COLORn]</td>
<td>Displays cell text using the corresponding color in the color palette, where n is a numeral that represents a color in the color palette. For example, [COLOR5] displays cell text in blue.</td>
</tr>
</tbody>
</table>

You can include the following currency symbols in a number format. Keep the ALT key pressed and type the ANSI code of the currency. The ANSI code should be followed by the format code for the number.
To type ANSI code for the currency symbol, turn on NUM LOCK and use the numeric keypad. As you type the ANSI code, the Custom box appears blank. The currency symbol is displayed only when you finish typing the code.

<table>
<thead>
<tr>
<th>Hold the ALT key down and type this code</th>
<th>To display</th>
</tr>
</thead>
<tbody>
<tr>
<td>0162</td>
<td>¢</td>
</tr>
<tr>
<td>0163</td>
<td>£</td>
</tr>
<tr>
<td>0165</td>
<td>¥</td>
</tr>
<tr>
<td>0128</td>
<td>C</td>
</tr>
</tbody>
</table>

### Conditional Symbols

You can apply conditional formatting to monitor the data in your report.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| [conditional value]     | Designates a different condition for each section. For example, data in a column has values ranging from 200 to 800 and you want the text “Poor” to be displayed in black for values less that 400, the text “Good” to be displayed in Red for values greater than 600, and the text “Average” to be displayed in blue for values ranging between 400 and 600. You can use the following code for these conditions: 
<400>[Black]”Poor”;
>600>[Red]”Good”;
[Blue]”Average”
In this example, [<400] and >600] are the conditional values. |
Custom Number Formatting examples

The following table lists examples of custom number formats. It includes the formatting symbols, the report data, and how that data is displayed after using the formatting.

<table>
<thead>
<tr>
<th>Format</th>
<th>Cell data</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>#.##</td>
<td>250.436</td>
<td>250.44</td>
</tr>
<tr>
<td></td>
<td>0.43</td>
<td>.43</td>
</tr>
<tr>
<td>#.0#</td>
<td>250.436</td>
<td>250.44</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>125.0</td>
</tr>
<tr>
<td>????.???</td>
<td>123.43, 45.90,</td>
<td>With aligned decimals</td>
</tr>
<tr>
<td></td>
<td>345.809</td>
<td></td>
</tr>
<tr>
<td>#.##0&quot;CR&quot;;#.#0&quot;DR&quot;;</td>
<td>2567</td>
<td>2,567CR</td>
</tr>
<tr>
<td></td>
<td>-4567</td>
<td>4,567DR</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>#.###</td>
<td>1500</td>
<td>1,500</td>
</tr>
<tr>
<td>0,</td>
<td>10,000</td>
<td>10</td>
</tr>
<tr>
<td>&quot;Sales&quot;=&quot;0.0&quot;</td>
<td>123.45</td>
<td>Sales=123.5</td>
</tr>
<tr>
<td>&quot;X=&quot;&quot;0.0;&quot;x=&quot;-0.0</td>
<td>-12.34</td>
<td>x=-12.3</td>
</tr>
<tr>
<td>&quot;Cust. No. &quot; 0000</td>
<td>1234</td>
<td>Cust. No. 1234</td>
</tr>
<tr>
<td>@[&quot;This Month&quot;]</td>
<td>Revenue</td>
<td>Revenue This Month</td>
</tr>
<tr>
<td>m-d-yy</td>
<td>2/3/03</td>
<td>2-3-03</td>
</tr>
<tr>
<td>mm dd yy</td>
<td>2/3/03</td>
<td>02 03 03</td>
</tr>
<tr>
<td>mmm d. yy</td>
<td>2/3/03</td>
<td>Feb 3, 03</td>
</tr>
<tr>
<td>mmmm d, yyyy</td>
<td>2/3/03</td>
<td>February 3, 2003</td>
</tr>
<tr>
<td>d mmmm yyyy</td>
<td>2/3/03</td>
<td>3 February 2003</td>
</tr>
<tr>
<td>hh&quot;h&quot; mm&quot;m&quot;</td>
<td>1:32 AM</td>
<td>01h 32m</td>
</tr>
<tr>
<td>h.mm AM/PM</td>
<td>14:56</td>
<td>2.56 PM</td>
</tr>
<tr>
<td>#?/?</td>
<td>1.25</td>
<td>1 1/4</td>
</tr>
<tr>
<td>#?/8</td>
<td>1.25</td>
<td>1 2/8</td>
</tr>
<tr>
<td>ALT+0163 ###</td>
<td>250.45</td>
<td>£ 250.45</td>
</tr>
<tr>
<td>#.###%</td>
<td>.08</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>2.8</td>
<td>280%.</td>
</tr>
<tr>
<td>$#* #.##0.00;$*</td>
<td>5632.567</td>
<td>$ 5,632.57</td>
</tr>
<tr>
<td>-#.#0.00</td>
<td>-12.34</td>
<td>$ -12.34</td>
</tr>
</tbody>
</table>
Formatting layers

Every report contains several different formatting layers, allowing you to retain control of how a report looks when it is pivoted or manipulated. You can ensure that the formatting continues to highlight the information that needs attention. There are two basic formatting layers—zones and grid units. Examples of zones are the rows headers and metric values of a report, while grid units are the values of a particular attribute or metric. The other formatting layers, such as thresholds and subtotals, can be thought of as extensions of these two basic types.
Zone formatting

The following diagram illustrates the basic *formatting zones* of a report. Each zone is formatted differently so that you can easily distinguish among them.

When data is manipulated in a report that is formatted by zone, the new location of the object determines what formatting is applied to it. For example, if you pivot Region from rows to columns in the preceding example, the background of the text changes from light grey to dark grey. It is now part of the column header, as shown below. The formatting of a zone does not move with the data.
Grid unit formatting

Grid units are the individual attributes, metrics, and consolidations that make up a report. Unlike zone formatting, grid unit formatting is attached to the object and moves with it when the object is pivoted. For example, the following report is the same as the previous examples, except that Region has been formatted, at the unit level. The header, that is, Region, is now black on light grey and the values (Northeast and Mid-Atlantic) are now black on white.

When Region is pivoted to the column area, as in the zone formatting example, the formatting accompanies the attribute. Compare the following example with the pivot example in the Zone formatting section.
Subtotals

Subtotal formatting can be applied to either zone or grid unit formatting. If the formatting is applied at the zone level, the formatting stays with that zone. If the formatting is applied at the grid unit level, when the unit is pivoted, the formatting moves with the unit.

In the following example, notice that the row subtotal formatting overwrites the column subtotal formatting.
Thresholds

Thresholds allow conditional formatting for metric values. It is similar to unit formatting because it is data-driven. For example, the following report has a threshold set for revenue less than $400,000.

<table>
<thead>
<tr>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>De Le Torre</td>
<td></td>
<td>$514,524</td>
<td>$391,121</td>
<td>$123,403</td>
</tr>
<tr>
<td>Kelly Laura</td>
<td></td>
<td>$329,888</td>
<td>$250,120</td>
<td>$79,768</td>
</tr>
<tr>
<td>Kiefer Jack</td>
<td></td>
<td>$399,888</td>
<td>$294,751</td>
<td>$95,137</td>
</tr>
<tr>
<td>Sawyer Leanne</td>
<td></td>
<td>$316,786</td>
<td>$240,110</td>
<td>$76,676</td>
</tr>
<tr>
<td>Sonder Melanie</td>
<td></td>
<td>$421,036</td>
<td>$318,975</td>
<td>$102,061</td>
</tr>
<tr>
<td>Yager Beth</td>
<td></td>
<td>$352,742</td>
<td>$275,208</td>
<td>$87,534</td>
</tr>
</tbody>
</table>

Besides the basic formatting options such as font and alignment, the cell contents can be replaced with any of the following when the condition is met:

- Replacement text, which is text, such as Good Sales.
- A replacement image. The destination of the image can be set using any of the following:
  - an absolute path (c:/images/img.jpg)
  - a URL to an image file (http://www.microstrategy.com/images/img.jpg)
  - a path on the local area network, which is in a UNC format (\machine_name\shared_folder\img.jpg)
  - a relative path from the document directory where the image is stored (images/img.jpg)
  - a relative path from \Document Directory where the image is stored

If you specify the location of the image as a directory rather than a URL, you must confirm that you can access the directory over the Web and the Desktop. If not, the image will not be displayed because it will not be available over the network. This problem of network access can be avoided by referencing a URL. If you specify the location of the threshold image as a UNC format, you cannot view threshold images when
you view the report in PDF format. This is because the Internet user account does not have permissions to a file on the network. Similarly, when the Intelligence Server is running on a system account, it does not have access to XSLs and HTML template files if the document directory is in a UNC format. In such cases also you cannot view threshold images when you view a report in the PDF format.

- A symbol chosen from a pre-defined list. In Web, these symbols are represented by an image file resembling the symbol used in Desktop.

**Order of layers**

With the different types of formatting, it is important that the interaction between them is clearly defined. How each of them impacts the final report display and in what order is crucial. Each succeeding layer overwrites the formatting of all its preceding layers. This is graphically illustrated in the following example.

The example is based on the **Basic Report** used earlier, which contains the Revenue metric. Use the Metric Editor to change the metric header to a bold, 12-point font. Wherever this metric is used, this header font is applied. Execute the report. The Revenue metric header appears the same as the other metric headers, because other formatting layers already set in the report overwrite the metric level formatting.

Italicize the column values and change the other font settings to default. Change the Revenue metric header to a white font. Since the Format menu in the Report Editor is used for this change, the new format applies to the current report only. The formatting dialogs for each are shown below.
The completed report looks like the following:

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>De Le Torre Sandra</td>
<td>$514,524</td>
<td>$391,121</td>
<td>$123,403</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kelly Laura</td>
<td>$329,888</td>
<td>$250,120</td>
<td>$79,768</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kieferson Jack</td>
<td>$389,888</td>
<td>$294,751</td>
<td>$55,137</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sawyer Leanne</td>
<td>$316,786</td>
<td>$240,110</td>
<td>$76,676</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sonder Melanie</td>
<td>$421,036</td>
<td>$318,975</td>
<td>$102,061</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yager Beth</td>
<td>$362,742</td>
<td>$275,208</td>
<td>$87,534</td>
<td></td>
</tr>
</tbody>
</table>
The final formatting for the Revenue metric header is a combination of the formats of the metric level header (set in the Report Editor), the column values, and the report metric. If all these formats were merged into one font dialog, it would look like the representation below.

![Settings dialog]

This dialog does not exist; it is presented only to further explain the example.

The following list describes all the formatting layers in the order that they are applied, starting with the first layer to be applied.

1. **Metric** specifies a format for the particular metric, regardless of the report it is on. This formatting, which can be applied to headers and values, is performed in the Metric Editor. Metric level formatting is overwritten by axis and all metrics formatting. Setting those layers to default allows the metric level formatting to display.

To format metric values at the metric level, the all metrics values formatting must be default. To use it for metric headers, set the axis headers formatting to default.

2. **Axis** formatting affects all the units of the axis. This zone formatting is overwritten by grid unit formatting. The axis formatting layers are located under the Rows and Columns options on the Format menu.

3. **Grid unit** allows you to format an individual report item, such as an attribute. It overwrites axis formatting. Every grid unit is listed on the Format menu.
4 **All metrics** formats the data zone, or where the metric values are displayed. It overwrites metric level formatting. The Format menu contains the All Metrics option.

5 **Report metric** formats an individual metric on a particular report. It does not change the formatting of the metric in other reports. Report metric formatting overwrites metric level and all metrics formatting. To format a metric at the report level, select the metric on the Format menu.

6 **Banding** enables row or column grouping by color to enhance readability. Banding formats are applied before subtotal formatting to allow subtotals to take priority. Select Grid, then Options to create banding.

7 **Column subtotals** formatting is overwritten by row subtotals when column and row subtotals intersect. Subtotal formatting can be applied as either zone or grid unit formatting, allowing you to select whether the formatting moves with the subtotals (grid unit) or is based on location (zone). To format subtotals as a zone, select Columns from the Format menu, then choose Subtotal headers or Values in the drop-down menu. Otherwise, select the grid unit from the Format menu.

8 **Row subtotals** formatting takes precedence over column subtotals when the two intersect. As with column subtotals, it can be applied as either zone or grid unit formatting.

9 **Report border** creates a border around the whole report.

    To set a report border, right-click the report but not a report object. Select Formatting, then Report Borders.

10 **Threshold** is the last layer applied so it overwrites all other layers.
The following table contains a matrix of each formatting layer and the layers that overwrite it.

<table>
<thead>
<tr>
<th>This layer...</th>
<th>Is overwritten by these layers...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric object - headers</td>
<td>Axis headers, grid unit headers, all metrics headers, report metric headers, column subtotal headers, row subtotal headers</td>
</tr>
<tr>
<td>Metric object - values</td>
<td>Axis values, grid unit values, all metrics values, report metric values, banding, column subtotal values, row subtotal values</td>
</tr>
<tr>
<td>Axis - headers</td>
<td>Grid unit headers, all metrics headers, report metric headers, column subtotal headers, row subtotal headers</td>
</tr>
<tr>
<td>Axis - values</td>
<td>Grid unit values, all metrics values, report metric values, banding, column subtotal values, row subtotal values</td>
</tr>
<tr>
<td>Grid unit - headers</td>
<td>All metrics headers, report metric headers, column subtotal headers, row subtotal headers</td>
</tr>
<tr>
<td>Grid unit - values</td>
<td>All metrics values, report metric values, banding, column subtotal values, row subtotal values</td>
</tr>
<tr>
<td>All metrics - headers</td>
<td>Report metric headers, column subtotal headers</td>
</tr>
<tr>
<td>All metrics - values</td>
<td>Report metric values, banding, column subtotals, row subtotals, threshold</td>
</tr>
<tr>
<td>Report metric</td>
<td>Banding, column subtotals, row subtotals, threshold</td>
</tr>
<tr>
<td>Banding</td>
<td>Column subtotals, row subtotals, threshold</td>
</tr>
<tr>
<td>Column subtotals</td>
<td>Row subtotals, threshold</td>
</tr>
<tr>
<td>Row subtotals</td>
<td>Threshold</td>
</tr>
<tr>
<td>Report border</td>
<td>None</td>
</tr>
<tr>
<td>Threshold</td>
<td>None</td>
</tr>
</tbody>
</table>
**Autostyles**

*Autostyles* provide predefined formatting styles, allowing you to standardize formatting among reports. Each autostyle is a collection of all the formatting layers, allowing you to format the different report sections. However, not every layer must be configured to create an autostyle. Default formatting values are supplied by the `guiprop.pds` file. For more information on defaults, see Appendix J, *Formatting Default Values*. Each formatting layer contains all the formatting properties:

- font
- alignment
- border
- pattern

HTML cannot display patterns as the background of a cell. Therefore, the patterns do not display in Web reports.

Notice that autostyles do not include number formatting. Numbers are usually formatted at a different level, such as the metric level. Retaining the report-level format allows your selected number format to remain.

Preconfigured autostyles are included in the Desktop, but you can create your own as well. If you do, design at the lowest layer possible, not at the grid unit level. If a formatted grid unit does not appear on a report, that formatting also does not appear on the report.

To deploy your own autostyles to users, simply place them in the Autostyles folder under the Public Objects folder. Other Desktop and Web users on the system can apply any autostyle saved in that location.

After an autostyle is placed in the folder, Web users cannot immediately access it. To refresh the autostyles list, Web users must log out and then log in.
So far, this chapter focused on report design—that is, the process of building reports from basic report components using the Report Editor in MicroStrategy Desktop or Web. You have learned how to design reports using the data definition and view definition. The data definition establishes how the data is accessed and manipulated in the data warehouse. It includes the report filter, Report Objects, and report limits. The view definition represents how the data is presented and manipulated in the Intelligence Server. Concepts such as the view template, formatting, thresholds, view filters, derived metrics, subtotals, and sorting make up the view definition.

Report design allows you to set up a controlled, user-friendly environment for report creators, who build reports from existing, predesigned reports in either Desktop or Web. Report creators can customize these reports with the wide range of powerful reporting functionality that report designers can make available.

Before outlining the steps to set up a report creation environment, let’s explore how a business intelligence project can possibly develop in terms of the complexity of reports and the experience of its users. The goal of project development is to unburden the report designer and the project administrator from participating in every report, by empowering users to find answers to their business questions independently. They are enabled to extract more meaning from the data and to work independently, rather than just view static report data.

The user categorizations in this section are based on the user privileges assigned in MicroStrategy.
At the beginning of a project, simple, non-interactive reports are set up by the report designer. Novice users execute the reports and view the data. Users who do not need or desire more complex analyses continue to run these types of reports throughout the life of the project. An example is a simple grid containing Region, Employee, and Revenue.

The first advance is the addition of some user interaction to the reports, such as prompts, drilling, and simple formatting. Defining drill paths allows a report designer to control the information a report creator can access, while providing options to explore and analyze the data. Prompting also provides a report creator with choices, but only those choices that the report designer has deemed appropriate and relevant. These privileges are given to a Web Reporter, although a report designer must set up the prompts and drill maps to ensure that the data definition is constructed correctly.

Drilling and prompting are described in more detail in succeeding chapters.

The next step is the Web Analyst level, where robust report customization is allowed. These options include pivoting, Report Objects access, derived metric creation, and view filter modification. The features available through these privileges allow a user to customize a report to answer a business question and then save the new report.

Reports can be designed with all objects in Report Objects and none on the grid. This provides a blank slate or foundation for report creators to customize and manipulate reports to suit their needs. Users who work with this type of report are Desktop Analysts. They cannot add or delete objects from the report but can alter what is viewed. In this step, report creators achieve independence from the report designers. Hence, they have the necessary tools to create customized reports in a safe environment where they are assured that the data makes sense and is relevant.

Finally, Web Professionals have increased access to all the objects in the project. They can enter the Design View to add and delete Report Objects, and create report filters. Report creation ends here, and report definition begins, because Web Professionals can modify the data definition without oversight.
What we have generically called a report designer in this chapter, is a combination of Web Professional and Desktop Designer. Desktop Designers develop new reports from scratch. They access various editors to create report components such as consolidations, custom groups, data marts, documents, drill maps, filters, metrics, prompts, and templates. The remainder of this book is aimed at Desktop Designers.

The following diagram depicts the project progression in terms of the user types.

As the project matures and the users’ experience increases, more advanced functionality is deployed to the user community. The Web Reporters begin with the simplest reports but as they become more comfortable with the product, they can become Web Analysts and Desktop Analysts, employing user interaction such as drilling and prompts. From the beginning, Desktop Designers and Web Professionals develop predesigned reports to deploy to less experienced users.
Privileges

The following graphic sums up the various privileges available in Desktop and Web. Not every user has to be granted all the privileges for a given role. For example, a Web Reporter may be allowed to execute reports only, without prompt and drilling options.

<table>
<thead>
<tr>
<th>Role</th>
<th>Privileges</th>
</tr>
</thead>
</table>
| **Web Reporter** | *Executes simple reports*  
                     *Answers prompts*  
                     *Drills on reports*  
                     *Performs some formatting* |
| **Web Analyst**  | Web Reporter privileges and:  
                        *Accesses Report Objects*  
                        *Creates derived metrics*  
                        *Modifies view filter*  
                        *Pivots reports*  
                        *Creates page by* |
| **Web Professional** | Web Analyst privileges and:  
                         *Uses Design View*  
                         *Adds & deletes Report Objects*  
                         *Modifies report filters* |
| **Desktop Analyst** | *Executes simple reports*  
                        *Answers prompts*  
                        *Drills on reports*  
                        *Formats reports, including creating auto styles*  
                        *Creates reports by manipulating Report Objects*  
                        *Creates derived metrics*  
                        *Modifies view filter*  
                        *Pivots reports*  
                        *Creates page by*  
                        *Sorts using advanced options* |
| **Desktop Designer** | *Designs new reports from scratch*  
                         *Creates report components such as:*  
                         - consolidations  
                         - custom groups  
                         - data marts  
                         - documents  
                         - drill maps  
                         - filters  
                         - metrics  
                         - prompts  
                         - templates  
                         *Uses Find and Replace and project documentation* |
A full discussion of privileges is included in the MicroStrategy System Administration Guide.

**Predesigned reports**

There are different ways to design reports for deployment to the report creation environment. A single project can use any, all, or none of these types of reports:

- static reports
- prompted reports
- Report Objects
- filter and template shortcuts

A single report can use more than one of these types. For example, a prompted report can have filter and template shortcuts.

**Static reports**

Static reports are basically reports without prompts. These reports are useful for inexperienced users or for delivering data to answer a specific query in a specific format.

**Prompted reports**

Prompted reports allow user interaction during report execution. A prompted report requires some input to finish the report content. The report definitions are dynamic, changing with each query when the information in the prompt dialog box is altered by a user.

Prompted reports can be as simple as choosing from a list of regions or as complicated as choosing all of the grid units and filtering conditions, such as the Report Builder and Report Wizard that are included in Desktop and Web. See Chapter 9, *Prompts*, for more information.
Report Objects

You can design a report that contains all of the relevant attributes and metrics to answer a particular category of business question, such as marketing. The objects are not placed on the grid, but on Report Objects. This allows report creators to use only those objects necessary for their specific analysis. The reports can be customized but the report creators are prevented from using incorrect data or calculations. These reports are used as foundations for report creators to build their own reports.

For example, reports have been set up in the MicroStrategy Tutorial for you to use when creating a new report. Create a new report by right-clicking the Desktop and selecting New, then Report. Notice the reports that are displayed on the New Grid dialog box. All these reports have been saved in the Object Templates folder to be made available for report creation. For more information on object templates, see Object templates.

To view the Object Templates folder, select Desktop Preferences from the Tools menu. Click Browsing Options. Select Display Hidden Objects, and click OK until you are returned to the Desktop. For more information on object templates, see Object templates.

Select Employee Analysis. No objects have been placed on the grid definition; they are all contained in Report Objects only. This provides users with a blank template so that they can create their own customized views of the report. Review the Report Objects—only those objects that are relevant to employees are included, such as hire date, employee name, and revenue.

Create another report, this time choose Time Analysis. Report Objects contains objects that are all relevant to time—month, year, quarter, and percent growth.
Shortcuts to filters and templates

The sample reports explained previously in this chapter have filters and templates that are created within a report and are known as embedded filters and templates. An *embedded filter* is generated when a filter is created on the fly in a report or when a copy of an existing filter is added to a report. Changes made to an embedded filter affect only the report in which it is contained because the filter exists only within that report. In contrast, a *shortcut to a filter* stands alone as a filter and can be used in multiple reports. When the filter is changed, the changes are propagated to all other reports that contain a shortcut to the same filter.

The difference between an embedded template and a shortcut to a template is similar to the difference between an embedded filter and a shortcut to a filter. An embedded template exists only within the context of a report, while a shortcut is linked to an existing template.

The diagram below illustrates the difference between embedded objects and shortcuts.

Separating the data definition of a report into a shortcut to an existing filter and a shortcut to an existing template helps make report deployment scalable and easily maintained. Details on shortcuts are included later in this chapter, in the *Shortcut to a filter* and *Shortcut to a template* sections.
Deploying predesigned reports

Choosing the type of predesigned reports to use is only one of the decisions a report designer makes when deploying reports. Other considerations are:

- predesigned report access
- object reuse
- caching
- privileges

Privileges have already been discussed in the previous section.

Accessing predesigned reports

To deploy these reports to users, you simply place them in a certain folder so that other users can access them. Reports saved in the Reports folder under the Public Objects folder can be viewed by other users on the system. Desktop users can navigate to reports in the Public Objects\Reports folder and execute them by double-clicking them. A Web user can navigate to the Shared Reports section and run those reports by clicking them.

You can also use the Reports folder under Object Templates to save reports that are frequently used to create new reports. They are displayed when a MicroStrategy Desktop user selects New, then Reports or a MicroStrategy Web user clicks Create New Reports.

To view the Object Templates folder, select Desktop Preferences from the Tools menu. Click Browsing Options. Select Display Hidden Objects, and click OK until you are returned to the Desktop. For more information on object templates, see Object templates.

Neither the deployment folders nor the reports are special, except that they are available for other users to access.
For more information on deployment, see the *Deploying Your Project* chapter of the *MicroStrategy Installation and Configuration Guide*.

**Reusing objects**

Shortcuts to filters and templates promote object reuse and good object management. For example, a project can have 50 reports that are all based on the same filter. When that filter has to be changed, how it was created is important. If the filter was built as a standalone object and implemented as a shortcut in the reports, the filter can be changed, and the alteration is applied to all of the dependent reports. However, if each report uses its own embedded filter, then that change must be duplicated in each of the 50 reports. Preventing this kind of object explosion and maintenance overhead is the basic justification behind object reuse.

Filters, of course, are not the only objects that can be reused. Templates, metrics, custom groups, and consolidations are among the objects that can be recycled.

When objects are used in multiple places, you can use the Search for Dependents tool to discover which objects contain them or other objects they contain. For example, you can search for all templates that contain the metric Profit or all metrics that are used by the template Store Sales.

**Caching**

In general, a *cache* holds recently accessed values to increase performance. For reports, a cache usually contains frequently requested reports, providing faster execution because then the reports do not access the data warehouse. For reports, the following caches are used:

- The report cache contains pre-processed report data and is stored on disk.
- The Intelligent Cube is identical to the report cache but is stored in the Intelligence Server memory. It allows manipulation of the data displayed in the report view.
The report view is an in-memory representation of the current view of a report, based on the view definition of that report. Each user running the same report has a unique report view on the Intelligence Server.

For more information on these caches, refer to the Intelligent Cubes section.

Intelligent Cubes do not need report re-execution for the following report modifications:

- drill
- pivot
- page-by
- sort
- subtotals
- outline mode
- banding
- report view format, such as changes to fonts and cell properties
- column alias
- add and remove report objects
- derived metrics
- view filter
- thresholds
- ranking

The traditional benefits of report caching include executing a report once and allowing many users to access the same report quickly from cache. Caching also improves database processing time. The new Intelligent Cube provides additional advantages that include the following:

- The response time for retrieving data and modifying the report should be almost immediate.
- Report caches can be created and refreshed on a scheduled basis during off-peak hours.
The report view does not need to display all the report objects available in the Intelligent Cube.

Objects can be moved between the report grid and Report Objects to allow ad hoc analysis within defined limits.

Multiple users can simultaneously have a unique representation of a report in the cube.

No additional SQL is generated when the Intelligent Cube contains all the necessary report objects. If a modification to a report needs additional information, SQL is automatically generated and submitted to the data warehouse.

Report definitions and views are stored in a central metadata repository, therefore any MicroStrategy user interface can easily share them.

**Shortcut to a filter**

When you add a filter to a report, you can

- Add it to the report filter. It is combined with any existing filters.

- Replace the report filter with a copy of the filter. Changes you make to the filter are not propagated to the original filter, and vice versa. This is also called a local or embedded filter and is the same as creating a filter on the fly in the report.

- Replace the report filter with a shortcut to the filter. Creating a shortcut to a filter allows you to use an existing filter on a report, taking advantage of the benefits of object reuse.

To choose from these options, right-click on a filter in the Object Browser.

In the Report Filter pane of the Design view, if the filter’s name is displayed and a shortcut icon appears in the title bar, it is a *shortcut to a filter*. This type of filter is sometimes referred to as a linked filter.
If you change the shortcut filter by, for example, removing an attribute, and then save the report, you can either create a local copy of the shortcut or retain the shortcut. If you create a copy, changes made to the filter in this report do not impact other reports. If you keep the shortcut, changes made to the filter in this report are propagated to all other reports that contain a shortcut to the same filter.

An example of a shortcut to a filter is included in *Shortcuts to a filter and a template example* below.

### Shortcut to a template

A *template* defines the layout of general categories of information in a report. A template specifies the information to retrieve from the data warehouse and how it is displayed in the report grid. Information on the template includes metric default values, banding options, join type setting, and data sorting options. You can create a stand-alone template in the Template Editor. Create a report-specific template using the Report Editor.

A *shortcut to a template*, which is sometimes referred to as a linked template, functions similarly to a shortcut to a filter.

When you add a template to a report, you can

- Replace the report template with a copy of the template. Changes you make to the template are not propagated to the original template. This is also called a local template and is the same as creating a template on the fly in the report.
- Replace the report template with a shortcut to the template. Creating a shortcut to a template allows you to use an existing template on a report.

To choose from these options, right-click on a template in the Object Browser.

In the Grid definition, if the template’s name is displayed and a shortcut icon appears in the title bar, it is a shortcut to a template.
If you change the shortcut template by, for example, adding a metric to Report Objects, and then save the report, you can either create a local copy of the shortcut or retain the shortcut. If you create a copy, changes made to the template in this report do not impact other reports. If you keep the shortcut, changes made to the template data definition in this report are propagated to all other reports that contain a shortcut to the same template. In the example, all those reports display the new metric in Report Objects.

**Shortcuts to a filter and a template example**

The following procedure creates a report using shortcuts to an existing filter and a template. The template places subcategories and revenue values by year on the report. The filter excludes April, May, and December from the metrics. When you are done, changes made to the shortcuts will impact other reports that use the same filter or template.

**To create a report with shortcuts to a filter and a template**

1. Create a new report.

2. Right-click the **Revenue Template**, which is found in the Supporting Objects folder. Select **Replace with shortcut to template**.

3. Right-click **Month Filter** in the same directory, and select **Replace Report Filter with a shortcut to this filter**.

4. Save the report. The Advanced Save Options dialog box opens.

5. Select **Retain the shortcut to the filter** and **Retain the shortcut to the template**. You can select to remember these settings for the next time you save a report.

6. Click **OK**.
When the report is executed, it looks like the following report sample.

<table>
<thead>
<tr>
<th>Subcategory</th>
<th>Year Metrics</th>
<th>2002 Revenue</th>
<th>2003 Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art &amp; Architecture</td>
<td></td>
<td>$ 5,334</td>
<td>$ 3,690</td>
</tr>
<tr>
<td>Business</td>
<td></td>
<td>$ 4,999</td>
<td>$ 4,923</td>
</tr>
<tr>
<td>Literature</td>
<td></td>
<td>$ 4,538</td>
<td>$ 4,204</td>
</tr>
<tr>
<td>Books - Miscellaneous</td>
<td></td>
<td>$ 742</td>
<td>$ 668</td>
</tr>
<tr>
<td>Science &amp; Technology</td>
<td></td>
<td>$ 3,615</td>
<td>$ 4,099</td>
</tr>
<tr>
<td>Sports &amp; Health</td>
<td></td>
<td>$ 2,280</td>
<td>$ 2,524</td>
</tr>
<tr>
<td>Audio Equipment</td>
<td></td>
<td>$ 540,950</td>
<td>$ 560,410</td>
</tr>
<tr>
<td>Cameras</td>
<td></td>
<td>$ 1,393,300</td>
<td>$ 1,210,090</td>
</tr>
<tr>
<td>Computers</td>
<td></td>
<td>$ 159,048</td>
<td>$ 142,092</td>
</tr>
<tr>
<td>Electronics - Miscellaneous</td>
<td></td>
<td>$ 297,764</td>
<td>$ 306,354</td>
</tr>
<tr>
<td>TV's</td>
<td></td>
<td>$ 442,444</td>
<td>$ 390,471</td>
</tr>
<tr>
<td>Video Equipment</td>
<td></td>
<td>$ 925,590</td>
<td>$ 874,490</td>
</tr>
</tbody>
</table>

This report is saved as **Shortcuts to Filter and Template**.

Now change the view definition of the report, which does not change either of the shortcuts. Move Year from the grid to Report Objects. Add a threshold of revenue greater than $1 million. Format the cell with bold font and 25% grey fill. The report is redisplayed as the following.

<table>
<thead>
<tr>
<th>Subcategory</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art &amp; Architecture</td>
<td></td>
<td>$ 9,024</td>
</tr>
<tr>
<td>Business</td>
<td></td>
<td>$ 9,922</td>
</tr>
<tr>
<td>Literature</td>
<td></td>
<td>$ 8,742</td>
</tr>
<tr>
<td>Books - Miscellaneous</td>
<td></td>
<td>$ 1,410</td>
</tr>
<tr>
<td>Science &amp; Technology</td>
<td></td>
<td>$ 7,714</td>
</tr>
<tr>
<td>Sports &amp; Health</td>
<td></td>
<td>$ 4,813</td>
</tr>
<tr>
<td>Audio Equipment</td>
<td></td>
<td>$ 1,101,360</td>
</tr>
<tr>
<td>Cameras</td>
<td></td>
<td>$ 2,603,390</td>
</tr>
<tr>
<td>Computers</td>
<td></td>
<td>$ 301,140</td>
</tr>
<tr>
<td>Electronics - Miscellaneous</td>
<td></td>
<td>$ 604,118</td>
</tr>
<tr>
<td>TV's</td>
<td></td>
<td>$ 832,915</td>
</tr>
<tr>
<td>Video Equipment</td>
<td></td>
<td>$ 1,800,080</td>
</tr>
</tbody>
</table>
This report is saved as **Shortcuts to Filter and Template with Thresholds**.

This does not change the linked template, because the attribute is removed from the view only; it remains on the Report Objects. The revenue is now calculated for all time, except for April, May, and December, which are excluded from the metric by the filter.

**Impact of modifying templates**

Recall how changes to a linked filter impact any dependent reports—if you create a local copy, changes made to the filter do not impact other reports. Alternatively, you can keep the shortcut, which allows changes made to the filter in this report to be propagated to all other reports that contain a shortcut to the same filter.

The effects of altering a template are more complex. For example, if a metric is removed from a template, the change can affect all, some, or none of the dependent reports. It depends entirely on how often the metric is included in the view definition of reports. The Template Dependency Validator allows you to conduct a quick impact analysis before saving any changes to a template.

The tool helps prevent the view definition from breaking the report because the view is asking for an object that is no longer in the data definition since it was removed from the underlying template. By alerting you to a potential problem, you can resolve the issue before any reports are affected. For example, a report has a shortcut to a template, which contains Country, Region, Metric 1, and Metric 2. The view filter is set to “Metric 1 > 20.” This is illustrated in the following diagram.
Suppose Metric 1 is removed from the template but not from the report view. When the report is executed, an error occurs because the view filter can no longer be evaluated (Metric 1 no longer exists).

When a template is modified and saved in Desktop, the Template Dependency Validator is triggered. The validator lists:

- reports that depend on the template
- reports that will not run if the change is completed

To resolve the problem, do one of the following:

- Cancel the change and re-evaluate.
- Open each dependent report and remove the dependencies, then change the template definition.
For the previous example, you could remove the view filter from the view definition of the dependent report.

The changes to the template are not saved until the Template Dependency Validator is closed. For more information on using this tool, see the online help.

Object templates

An **object template** allows you to use a predefined structure to begin creating a new object. For example, you may want to create many filters that contain the current month as part of the definition. Having a filter object template that contains the current month allows you to skip the step of defining that part of the filter each time you create a filter. In other words, you only have to define that filtering condition once. When you use the filter object template, you automatically have the current month condition in every new filter you create.

Another example is a need to build multiple reports containing the attribute Day and the metrics Revenue, Cost, and Profit. To reduce the time spent creating these similar reports, define a report with these objects and save it in the Object Templates folder, thus creating a report object template.

To be used as an object template, the object must be saved in the Object Templates folder. This is the only difference between an object and an object template (like a report and a report object template).

You can create object templates for the following objects:

- consolidations
- custom groups
- filters
- metrics
- reports
- templates
In Desktop Preferences, you can determine, for each type of object, whether to be prompted for an object template when creating a new object.

If an object template is altered, the change is not propagated to previously defined objects.

Do not confuse object templates with templates. A template defines the layout of general categories of information in a report. It specifies the information to retrieve from the data warehouse and the way you want it to be displayed in the Grid view of reporting. A template does not include filters, while object templates can contain filters. Combine a template and a filter to create a report. An object template is already a report and could be run as is, without modifications. An object template is equivalent to a template in Microsoft Word, which defines templates as a special kind of document that provides basic tools for shaping a final document. An object template does not have to be a report; it can be a metric, filter, or other object as described previously.

**Empty object templates**

Empty object templates are a subset of object templates. The only difference between the two is that object templates contain a definition and empty object templates do not.

Empty object templates allow you to set default formatting and other properties on a project level for new reports, templates, and metrics. This helps you control new objects created in your project. For example, you can create an empty metric object template with currency formatting or an empty report object template set to outline mode. Notice that the empty object template contains properties only, not a “definition”—that is, empty metric object templates do not have formulas, empty report object templates do not include attributes, metrics, or other grid units in the report objects.

Empty object templates are saved in the Object Templates folder since they can be used only as a template.
Why would you use an empty report object template? You may have a set of properties that should be applied to each new report, but you do not want to define those properties for each report. For example, your project has a series of reports that must be exported in Excel format to a particular location. A specific Excel macro must be run after the report is exported. You can create an empty report object template, called Excel Export Settings, with these specifications. When the Excel Export Settings report is used to create a new report, the new report contains the correct information.

An empty metric object template contains formatting and other properties but does not include formulas. Like empty report object templates, they can be used as a starting point for creating objects with specific preset properties. For example, a project requires all currency values to include cents for precision and to distinguish negative values in red font. To meet these conditions, create an empty metric object template named Currency Formatting and set these formats. When a user creates a new metric that returns currency values, he selects Currency Formatting in the New Metric dialog box. The formatting for red negative values and cents is included in the new metric.

The properties available in an object template vary with the type of object.

- An **empty metric object template** does not contain a formula but can contain the following properties:
  
  - formatting properties
  
  - VLDB properties
  
  - formula join type
  
  - metric join type
  
  - metric column properties
• An **empty report object template** does not contain any grid units, that is, attributes, metrics, consolidations, and so on. An empty report contains:

  – export options
  – filters
  – formatting
  – grid options
  – report data options
  – VLDB properties
  – other settings such as merge header cells and automatic column width

• An **empty template object template** is similar to an empty report object template.

You can also set a project default for the empty object templates. This means that if a user chooses **Do not show this window again** in the New Grid or New Metric dialog box, the default object template is used. This setting is found in project configuration.

Another project configuration setting, **Show empty template**, controls whether the object template named Empty Report, Empty Template, or Empty Metric is displayed. When a user selects one of these object templates in the New Grid or New Metric dialog box, the object template does not really exist, as an object. Instead, it is created dynamically using a set of default properties. In contrast, when you use an object template, even if it is empty, it is a static object. Regardless of the Show empty template setting, your object templates are displayed.
Evaluation order

The order in which data is calculated affects the report data set. By using evaluation order settings, you can control the order in which compound smart metrics, consolidations, derived metrics, report limits, and subtotals are calculated and resolved for a given report.

Evaluation order is the order in which different kinds of calculations are performed by the Analytical Engine. The following simple example illustrates how evaluation order can influence the report data set.

<table>
<thead>
<tr>
<th>States Consolidation</th>
<th>Revenue</th>
<th>Cost</th>
<th>Revenue/Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>$15</td>
<td>$10</td>
<td>1.5</td>
</tr>
<tr>
<td>Virginia</td>
<td>$20</td>
<td>$15</td>
<td>1.33</td>
</tr>
<tr>
<td>New York + Virginia</td>
<td>$35</td>
<td>$25</td>
<td>X</td>
</tr>
</tbody>
</table>

In the above example, two calculations are involved, the States Consolidation and the compound smart metric Revenue/Cost.

The following conditions apply:

- If the States Consolidation is calculated first, X represents the Revenue/Cost value of the last row, and the result is 35/25 or 1.4.
- If Revenue/Cost is calculated first, X represents the Revenue/Cost value of the last column, and the result is 1.5 + 1.33 or 2.83.

By default, the compound smart metric is evaluated before the consolidation, yielding a result of 2.83. The next section, Default evaluation order, provides more detail.
Default evaluation order

The default order of calculation (the MicroStrategy 6.x evaluation order) is as follows:

1. compound smart metrics

2. consolidations, which are evaluated by their relative position on the report template:
   - rows, from left to right
   - columns, from top to bottom

For examples of how evaluation order affects consolidations, see the following:


- *Consolidation and view evaluation order example* in this chapter demonstrates the interaction of a consolidation, subtotals, and the view definition with the evaluation order.

3. report limits

4. subtotals

Compound metrics that are not the direct aggregations of other metrics can be used in the evaluation order by setting the Allow Smart Metrics option of the Metric Editor to **Yes**.

Page-by and sorting are determined last, to arrange the positions of the calculation results. Their evaluation order cannot be changed.
Specified evaluation order

You can specify the evaluation order by assigning a calculation a positive number to indicate the order in which it is to be calculated. When handling calculations, MicroStrategy first performs those to which default rules of order apply, then those that have been assigned a number. Use the Report Data Options dialog box to specify the evaluation order. The setting is found under Calculations, then Evaluation Order.

The MicroStrategy 6.x evaluation order described in Default evaluation order permits you to reorder consolidations only. Changing from this setting allows you to alter the evaluation order for the following objects:

- compound smart metrics
- consolidations
- derived metrics
- metric limits
- subtotals

A compound metric that has not been identified as smart cannot be part of a specified evaluation order; it is always calculated first, as discussed in Default evaluation order.

Evaluation order in data definition and view definition

The data set evaluation order settings control the order of evaluation for consolidations, report limits, compound smart metrics, and subtotals. Once the view definition becomes different from the data definition, the View evaluation order is activated in the Report Data Options: Evaluation Order dialog box. It becomes available when:

- objects are on the Report Objects but not on the grid
- a view filter is defined
- a derived metric is used
These actions must include or affect objects on which you can set the evaluation order. For example, a consolidation on the Report Objects but not on the grid will activate the View evaluation order. Since a derived metric is, by definition, a compound smart metric, it always activates the View evaluation order.

A separate view order is necessary because any manipulation of the view that does not change the SQL is performed after the base report is completed. Therefore, objects in the Data Set evaluation order settings are evaluated first, and then those in the View evaluation order are evaluated. The following table describes where objects are evaluated.

<table>
<thead>
<tr>
<th>Object</th>
<th>Evaluation Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consolidation</td>
<td>Data Set</td>
</tr>
<tr>
<td>Derived metric</td>
<td>View</td>
</tr>
<tr>
<td>Report limit</td>
<td>Data Set</td>
</tr>
<tr>
<td>Compound smart metric</td>
<td>Data Set or View:</td>
</tr>
<tr>
<td></td>
<td>• Set automatically to View</td>
</tr>
<tr>
<td></td>
<td>• Can be changed manually</td>
</tr>
<tr>
<td>Subtotals</td>
<td>View</td>
</tr>
</tbody>
</table>

### Data set vs. view evaluation order example

Consider the following report, where Sales Rank is a smart metric ranking the Revenue metric. Eight rows of data are returned.

<table>
<thead>
<tr>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Sales Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$ 2,334,864</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>$ 3,413,340</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Southeast</td>
<td>$ 2,016,186</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Central</td>
<td>$ 1,773,270</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>South</td>
<td>$ 1,380,991</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Northwest</td>
<td>$ 1,485,182</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Southwest</td>
<td>$ 2,816,334</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Web</td>
<td>$ 1,716,267</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
To restrict the report, create a metric limit of Revenue greater than $2,000,000. Only those Regions with a Revenue greater than $2,000,000 will appear on the report. Force Sales Rank to be calculated before the metric limit, as described in the following procedure.

**To change evaluation order**

1. From the Data menu, choose **Report Data Options**. The Report Data Options dialog box opens.

2. Under Categories, expand **Calculations** and then select **Evaluation Order**. Calculations – Evaluation Order appears on the right side of the dialog box.

3. Clear the **Show consolidations only** check box. Now objects other than consolidations are also displayed in the evaluation order boxes.

Notice that both the Sales Rank metric and the report limit are calculated in the data set. The view does not differ from the data set, so nothing can be calculated in the view.

4. Click in the Evaluation Order column of Sales Rank and select 1 from the pull-down list.

5. Click in the Evaluation Order column of Report Limit and select 2 from the pull-down list.

6. Click **OK** to return to the report.

As shown below, only four rows are displayed now. Notice that the values for neither Revenue nor Sales Rank have changed. The report has not changed, but the amount of data displayed has.

<table>
<thead>
<tr>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Sales Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td></td>
<td>$2,334,864</td>
<td>6</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td></td>
<td>$3,413,340</td>
<td>8</td>
</tr>
<tr>
<td>Southeast</td>
<td></td>
<td>$2,016,186</td>
<td>5</td>
</tr>
<tr>
<td>Southwest</td>
<td></td>
<td>$2,816,334</td>
<td>7</td>
</tr>
</tbody>
</table>
Now, create a derived metric, which causes the report view to differ from the data set.

**To create a derived metric**

1. Select **New Metric** from the Insert menu. The Input Metric Formula dialog box opens.

2. Double-click **Revenue** in the list of Report Objects. It appears in the metric definition on the right.

3. Type `/2`, which also appears in the metric definition.

4. Click **OK** to return to the report.

You receive the following error:

> The functionality/report manipulation you are trying to use is incompatible with the report objects Evaluation Order settings.

This occurs because once a derived metric is added to a report, the view evaluation order is activated. As a smart metric, the Sales Rank metric is automatically moved to the view evaluation order, while the report limit stays in the data set. Recall that objects in the view are calculated after objects in the data set are computed. Sales Rank is set to be calculated first, but because it is in the view evaluation order, it cannot be evaluated until after the data set is complete. This conflict triggers the error message.

To resolve the conflict, you can set Sales Rank back to the default evaluation order, add the derived metric, and then set Sales Rank to be evaluated first in the data set. The report then looks like the following:

<table>
<thead>
<tr>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Revenue/2</th>
<th>Sales Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$ 2,334,864</td>
<td>$1,167,432</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>$ 3,413,340</td>
<td>$1,706,670</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Southeast</td>
<td>$ 2,016,185</td>
<td>$1,008,093</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Southwest</td>
<td>$ 2,816,334</td>
<td>$1,408,167</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
Notice that the values for neither Revenue nor Sales Rank have changed from the initial report. If you display the report after adding the derived metric but before moving the Sales Rank calculation to the data set, Sales Rank is calculated based only on the four rows displayed. That is, Southeast is ranked one, Northeast two, and so on.

**Consolidation and view evaluation order example**

The following example demonstrates how the default evaluation order and view evaluation order affect consolidations. For another example on consolidations, see *Evaluation order* in Chapter 8, *Custom Groups and Consolidations*. That example presents a report containing two consolidations but does not discuss the view evaluation order.

You want to compare the revenues for the Years 2002 and 2003 by Category and Subcategory. First, create a consolidation called Years to calculate the difference. The consolidation elements are defined below.

- 2002: Year = 2002
- 2003: Year = 2003
Build a report with Category and Subcategory as the rows. Place the Years consolidation and the Revenue metric on the columns, then enable subtotals. When you execute the report, the following is displayed.

<table>
<thead>
<tr>
<th>Category</th>
<th>Subcategory</th>
<th>Years Metrics</th>
<th>2002 Revenue</th>
<th>2003 Revenue</th>
<th>Percentage Difference Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Books</td>
<td>Art &amp; Architecture</td>
<td></td>
<td>$ 7,849</td>
<td>$ 6,136</td>
<td>27.92%</td>
</tr>
<tr>
<td>Books</td>
<td>Business</td>
<td></td>
<td>$ 7,510</td>
<td>$ 7,487</td>
<td>1.54%</td>
</tr>
<tr>
<td></td>
<td>Literature</td>
<td></td>
<td>$ 7,103</td>
<td>$ 6,602</td>
<td>7.59%</td>
</tr>
<tr>
<td>Books</td>
<td>Books - Miscellaneous</td>
<td></td>
<td>$ 1,181</td>
<td>$ 1,000</td>
<td>18.10%</td>
</tr>
<tr>
<td></td>
<td>Science &amp; Technology</td>
<td></td>
<td>$ 6,093</td>
<td>$ 5,955</td>
<td>2.32%</td>
</tr>
<tr>
<td></td>
<td>Sports &amp; Health</td>
<td></td>
<td>$ 3,393</td>
<td>$ 3,612</td>
<td>(10.99%)</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$ 33,229</td>
<td>$ 30,992</td>
<td>46.57%</td>
</tr>
<tr>
<td>Electronics</td>
<td>Audio Equipment</td>
<td></td>
<td>$ 852,990</td>
<td>$ 843,270</td>
<td>1.15%</td>
</tr>
<tr>
<td>Electronics</td>
<td>Cameras</td>
<td></td>
<td>$ 2,237,510</td>
<td>$ 1,858,920</td>
<td>20.37%</td>
</tr>
<tr>
<td>Electronics</td>
<td>Computers</td>
<td></td>
<td>$ 253,213</td>
<td>$ 219,647</td>
<td>15.28%</td>
</tr>
<tr>
<td></td>
<td>Electronics - Miscellaneous</td>
<td></td>
<td>$ 486,748</td>
<td>$ 469,214</td>
<td>3.74%</td>
</tr>
<tr>
<td></td>
<td>TV's</td>
<td></td>
<td>$ 707,093</td>
<td>$ 612,099</td>
<td>15.52%</td>
</tr>
<tr>
<td></td>
<td>Video Equipment</td>
<td></td>
<td>$ 1,415,430</td>
<td>$ 1,345,520</td>
<td>5.20%</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$ 5,952,984</td>
<td>$ 5,348,670</td>
<td>61.25%</td>
</tr>
</tbody>
</table>

Due to space constraints, this report sample contains a report filter to include the Books and Electronics categories only.

Notice that the Percentage Difference is calculated correctly for each Subcategory, as shown in the following example:

\[
\frac{\text{Revenue 2002} - \text{Revenue 2003}}{\text{Revenue 2003}} = \frac{\$7,849 - \$6,136}{\$6,136} = 27.92\%
\]

The totals for the Revenue column are also correct. However, the totals for the Percentage Difference column are wrong:

\[
\frac{\text{Total Revenue 2002} - \text{Total Revenue 2003}}{\text{Total Revenue 2003}} = \frac{\$33,229 - \$30,992}{\$30,992} = 7.22\%
\]

The report calculates the Percentage Difference as 46.57%. Why? The answer lies in the default evaluation order, which calculates the consolidation before the total. In other words, the total is determined by adding all the values in the column, and not by the correct method applicable here of calculating the formula \([(\text{2002} - \text{2003})/\text{2003}]\) across the rows. To calculate the total across the rows, change the evaluation order to calculate the total before the consolidation.
For instructions, see *To change evaluation order*. Set Total to 1 and Years to 2.

The report is redisplayed as shown below:

<table>
<thead>
<tr>
<th>Category</th>
<th>Subcategory</th>
<th>Years Metrics</th>
<th>2002 Revenue</th>
<th>2003 Revenue</th>
<th>Percentage Difference Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Books</td>
<td>Art &amp; Architecture</td>
<td>$ 7,649</td>
<td>$ 6,136</td>
<td>27.92%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Business</td>
<td>$ 7,610</td>
<td>$ 7,437</td>
<td>1.64%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Literature</td>
<td>$ 7,103</td>
<td>$ 6,602</td>
<td>7.59%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Books - Miscellaneous</td>
<td>$ 1,181</td>
<td>$ 1,000</td>
<td>18.10%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Science &amp; Technology</td>
<td>$ 6,093</td>
<td>$ 5,955</td>
<td>2.32%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sports &amp; Health</td>
<td>$ 3,393</td>
<td>$ 3,812</td>
<td>(10.99%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$ 33,229</td>
<td>$ 30,992</td>
<td>7.22%</td>
<td></td>
</tr>
<tr>
<td>Electronics</td>
<td>Audio Equipment</td>
<td>$ 852,990</td>
<td>$ 843,270</td>
<td>1.15%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cameras</td>
<td>$ 2,237,510</td>
<td>$ 1,858,920</td>
<td>20.37%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Computers</td>
<td>$ 253,213</td>
<td>$ 219,647</td>
<td>15.28%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Electronics - Miscellaneous</td>
<td>$ 486,748</td>
<td>$ 469,214</td>
<td>3.74%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TV's</td>
<td>$ 707,093</td>
<td>$ 612,099</td>
<td>15.52%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Video Equipment</td>
<td>$ 1,415,430</td>
<td>$ 1,345,520</td>
<td>5.20%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$ 5,952,984</td>
<td>$ 5,348,670</td>
<td>11.30%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$ 5,986,213</td>
<td>$ 5,379,662</td>
<td>11.27%</td>
<td></td>
</tr>
</tbody>
</table>

While this solution works for this particular instance of the report, what happens when you move Subcategory to the Report Objects? You cannot do it because you receive an error that the manipulation is incompatible with the evaluation order settings.

Moving Subcategory to the Report Objects activates the view evaluation order. When this occurs, the Years consolidation stays in the data set evaluation order and the subtotals move to the view evaluation order. Since the data set is evaluated before the view, the consolidation is calculated before the subtotals. However, you set the subtotals to be calculated first, to produce the correct results for the report. This conflict causes error.

To produce the correct results, you must either delete Subcategory from the report completely (that is, not move it to the Report Objects) or create a second report without Subcategory.
Find and replace

Find and replace allows you to globally change reports, templates, and metric formatting. You can replace autostyles, Report Data Options, metric formatting, and graph font for a set of reports, templates, and metrics.

This feature is helpful when a report design changes. For example, suppose the formatting of all your inventory reports must be changed. Simply create a new autostyle with the changes, search for only inventory reports, and substitute the new autostyle. Find and replace autostyles allows you to quickly switch the formatting of the selected reports.

The feature also is useful when standardizing the appearance of a set of metrics across the project. However, changes to metric formatting do not overwrite the format of such metrics set by individual reports or templates.

For detailed instructions and a description of the Find and Replace interface, see the online help.

Note the following:

- Before you can use the Find and Replace interface, you must have the **Use Find and Replace** privilege. For more information on privileges, see the *MicroStrategy System Administration Guide*.

- Replacing autostyles or Report Data Options invalidates the report caches. A new cache is created when the report is run for the first time after the replacement.
**Autostyles**

Find and replace autostyles allows you to apply an autostyle to a selected set of reports and templates, easily creating a standard format for the selected objects.

Note the following:

- A report (or template) records the ID of the style when the style is applied, but it is possible that the style was changed after it was applied. In this case, the report is using the Custom style because the definition has changed. When Find and Replace is run, the report using the target autostyle, even if it has been changed and is displayed in the Report Editor as Custom, will be updated with the new autostyle.

- If the replacement autostyle is not saved in the Autostyles or My Objects folders, the autostyle name is displayed as Custom in the report, although the correct autostyle is applied.

**Report Data Options**

Find and replace allows you to modify the Report Data Options for a set of reports and templates. Since many of the Report Data Options, such as evaluation order, are relevant for only the report/template in which they are defined, globally changing these can result in unexpected outcomes. Therefore, only the following Report Data Options are available in Find and Replace:

- **Display: Null Values**, which allows you to temporarily replace null values during report calculation or display.

- **General: Drilling**, which allows you to enable drilling for the report/template and set various drilling options.

- **General: Advanced**, which allows you to save the page-by-settings on a report template. It also allows you to save any modifications that you make to a prompted report, after executing that report.
When a report with a linked template is changed, a dialog box appears, prompting you to break the link. If the link is broken, the changes are not propagated to the standalone template and a new, embedded template is used. If the link is not broken, the linked template is changed.

**Metric Formatting**

Find and Replace allows you to modify the formatting of a selected set of metrics. You can search for specified sets of metrics, select metrics according to search criteria, or select all metrics in the project. Once you’ve determined the set of metrics you want to work with, you can replace their header format, value format, or both, using the format of an existing metric or the Format Cells dialog box.

**Graph Font**

Find and Replace allows you to apply a font of your choice to graph titles and labels in selected sets of reports and templates. The default graph font is Arial, and you can change it to any other font available in the font drop-down list. You may have to change the default graph font when using some international character sets. For example, if the character set is set to Turkish in Web, you should select the font other than Arial to display correctly the graph title and labels.

Graph fonts can be changed in the following places:

- **For new graphs**: Change the default graph font in Project Configuration, Report Definition, Graph.
- **For existing graphs**: Change the default graph font in Find and Replace, Select, Graph Font.
The bulk export feature allows you to select a report and export it to a delimited text file. Using this feature, you can retrieve result sets from large datasets without having to load the datasets in memory. As the result sets are brought back incrementally, no limit is set on the size of the dataset to be retrieved.

While you can schedule a bulk export report for execution on the Web through Narrowcast Server, you cannot execute a bulk export report on Desktop (when you right-click a bulk export report, the Run Report option is not available). You can, however, view the report in Design View and SQL View on Desktop, since all the other views are disabled.

To execute a bulk export report on the Web using Narrowcast Server, you must have the “Web execute bulk export report” privilege. However, you must have selected the report for bulk export on Desktop first, which requires the “Use report data options” privilege.

### To select a report for bulk export

1. On Desktop, open a report in Report Editor.
2. From the Data menu, select Report Data Options. The Report Data Options dialog box is displayed.
3. In the General category, select Advanced.
4. Select the Set as Bulk Export report. Execute from Web using Scheduled File Export enabled by Narrowcast Server check box.

After you specify a report as a bulk export report, its icon on Desktop changes as shown in the following figure.
Note the following:

- If you open a bulk export report in SQL view, certain column names are replaced by column aliases. You can view the column names by changing the default VLDB properties for metrics. To do this, from the Project Configuration Editor, select **Database instances** in the Project Definition category, then click **VLDB Properties**, and then select the **Default To Metric Name** option from the Metrics folder. Select the **Use the metric name as the default metric column alias** option, and save the changes. Restart the MicroStrategy Intelligence Server. For detailed information on VLDB Properties, see the *MicroStrategy System Administration Guide*.

- If you have selected multiple reports, including bulk export reports, and choose the Run Report option from the right-click menu, all the bulk export reports are filtered out and are opened in the Report Editor; the other reports are executed as usual.

- Bulk export reports cannot be used or dragged into HTML documents or Report Services documents.

- A report that uses the bulk export feature has the same restrictions as a data mart report.
Introduction

The Freeform SQL functionality adds great flexibility to MicroStrategy’s query and reporting capabilities. Traditionally, you use the MicroStrategy Engine to generate SQL to run against one specific relational database to get a desired report. Starting from 8.0, in addition to generating reports in the traditional way, you can also use your own customized SQL statements to generate reports from operational systems included in a MicroStrategy project. This capability could save you a tremendous amount of work time since you do not need to place the data into a data mart or data warehouse first.
Creating Freeform SQL Reports

This chapter discusses different aspects of the Freeform SQL functionality, including the following:

- Freeform SQL reporting (on page 136)
  - When should I use the Freeform SQL reporting feature? (on page 136)
  - SQL query syntax (on page 137)
  - SQL support ((on page 138)
  - Freeform SQL reports vs. standard reports (on page 140)
  - Freeform SQL reports in Report Services documents (on page 141)

- Reporting features (on page 143)
  - Filters (on page 143)
  - Prompts (on page 143)
  - Drilling (on page 147)

- Security for data access (on page 148)
  - Access control list (on page 148)
  - Security filters (on page 149)

- Managed objects (on page 153)

- Creating Freeform SQL reports (on page 158)
  - Creating a Freeform SQL report from a database (on page 158)
  - Creating a Freeform SQL report from an Excel file (on page 160)
  - Creating a Freeform SQL report from a text file (on page 163)
  - Creating a Freeform SQL report using a stored procedure (on page 166)
The following graphic is an illustration of the Freeform SQL Editor, where you define the SQL statement for the report. Notice the different panels for different purposes. For details about these panels, please refer to the online help (search for the “Using the Freeform SQL Editor” topic).
Freeform SQL reporting

The Freeform SQL reporting feature allows you to use your own SQL statements to access data from various data sources, including relational databases, Excel files, and flat files, as long as they are included in the MicroStrategy environment. Details on how to create Freeform SQL reports from these data sources are discussed in this chapter as well as in the online help (search for the “Creating Freeform SQL reports” topic).

How to use the Freeform SQL reporting feature effectively depends on your work environment. As with every other MicroStrategy functionality, before you start using this feature, you need to assess your particular work situation and find a way to strike a good balance between project maintainability and fast-paced development. For example, building three or four Freeform SQL reports could be very valuable, but building 100 such reports could make maintenance and testing very difficult.

Whether you should use the Freeform SQL reporting feature at all is another question that you should ask yourself. Most likely, you may want to consider using this feature if you are in one of the situations discussed as follows.

When should I use the Freeform SQL reporting feature?

If your company is used to creating static reports using customized SQL to retrieve data from a certain data source, and especially if your SQL queries have worked well in the past, then you may want to simply use MicroStrategy to deploy those reports to your users. There is no need to recreate the SQL with the MicroStrategy Engine, as is done if the data is moved to a data warehouse for report generation.
In the same spirit, if you have existing stored procedures that have proven to be successful, then you can continue to use them to generate MicroStrategy reports. One important thing to note is that you must know what exact data the stored procedure is supposed to retrieve because this information is essential in building a Freeform SQL report. Specifically, you need to know the number of columns, column names, and their data types, all of which are necessary for mapping the columns to MicroStrategy objects. For examples on different databases, please see the online help topic “Creating Freeform SQL reports using stored procedures”.

Another situation for which you might want to use Freeform SQL reporting is when you need to run queries against a set of OLTP tables that are not set up for OLAP analysis. As for all the Freeform SQL reports, connecting to the right data source is a prerequisite.

**SQL query syntax**

Well-written SQL queries are the key to building successful Freeform SQL reports. To take full advantage of the Freeform SQL reporting feature, MicroStrategy recommends that you ensure the correctness and validity of your SQL statements before creating any such reports. MicroStrategy does not offer consultation or technical support for the syntax or composition of your SQL queries.

Depending on your needs, you can compose SQL statements in several ways:

- create new SQL statements from scratch, if you have not created any before
- use existing SQL statements that you previously defined, which have proven to be successful in terms of retrieving data from the data source
- tune the MicroStrategy Engine-generated SQL by modifying it to suit your needs

This means you can reuse the Engine-generated SQL by changing some of its clauses or syntax to get a different result set.
Note that you cannot tune the Engine-generated SQL that involves the use of the Analytical Engine. Typically, the Analytical Engine comes into play for metric qualification using analytical functions (such as OLAP functions), custom group banding, use of the plug-ins, and so on. If the Analytical Engine is used for any part of the SQL during the report generation, that part is labeled as “[An Analytical SQL]” in the report SQL.

All MicroStrategy functions, including the ones that use the Analytical Engine, are described in detail in the *MicroStrategy Functions Reference*.

**SQL support**

With the Freeform SQL reporting feature, you can use both single-pass and multi-pass SQL queries to generate reports. Make sure that you use derived table or common table expression syntax.

If you have to use derived table or common table expressions, then you can have only one SELECT clause in your SQL query. This means that a report SQL statement with Pass 1, Pass 2, Pass 3..., as can be found in many MicroStrategy Tutorial reports, cannot yield the desired report, unless you modify the query using derived tables or common table expressions. Please check the database that you use to ensure that derived tables or common table expressions or both are supported.

For example, if you have the following multi-pass SQL statement, it needs to be converted to derived table or common table expression syntax (also shown in the following).
Multi-pass SQL

create table MQ00(
    A1 INTEGER,
    B1 INTEGER)
insert into MQ00
select a11.[A1] AS A1,
    a11.[B1] AS B1
from [T1] a11
where (a11.[M1] > 0);

select a11.[A1] AS A1,
    a11.[B1] AS B1,
    a12.[B2] AS B2,
    a11.[M1] AS M1
from [T1] a11,
    [MQ00] pa1,
    [T2] a12
where a11.[A1] = pa1.[A1] and
    a11.[B1] = pa1.[B1] and
    a11.[B1] = a12.[B1]

drop table MQ00

Derived table

select a11.[A1] AS A1,
    a11.[B1] AS B1,
    a12.[B2] AS B2,
    a11.[M1] AS M1
from [T1] a11,
    (select a11.[A1] AS A1,
        a11.[B1] AS B1
from [T1] a11
where a11.[M1] > 0) pa1,
    [T2] a12
where a11.[A1] = pa1.[A1] and
    a11.[B1] = pa1.[B1] and
    a11.[B1] = a12.[B1]
Common table expression

with pa1 as
  (select a11.[A1] AS A1,
   a11.[B1] AS B1
  from [T1] a11
  where a11.[M1] > 0)
select a11.[A1] AS A1,
   a11.[B1] AS B1,
   a12.[B2] AS B2,
   a11.[M1] AS M1
from [T1] a11,
   pa1,
   [T2] a12
where a11.[A1] = pa1.[A1] and
   a11.[B1] = pa1.[B1] and
   a11.[B1] = a12.[B1]

Freeform SQL reports vs. standard reports

You can create Freeform SQL reports using your own SQL queries against a data warehouse, or from an Excel file, or a flat file (information on how to create these reports is provided later in this chapter). Although Freeform SQL reports can only be created on Desktop, once created, they can be executed from both Desktop and the Web like any other standard reports. Functions that apply to the execution and manipulation of standard reports also apply to Freeform SQL reports, including the following:

- formatting
- exporting
- thresholds
- graphing
- Narrowcast Server subscriptions and report execution
- object security
- OLAP services
- prioritization
• Report Services documents
• scheduling
• subtotals
• Web report execution

However, the following features do not apply to Freeform SQL reports:
• custom group
• consolidation
• transformation
• filter
• save as template/filter
• report as filter
• data marting

**Freeform SQL reports in Report Services documents**

Once created, Freeform SQL reports can be included in Report Services documents in the same way as standard reports. A document can also contain reports from other sources, such as OLAP cube reports. For information regarding OLAP cube reports, please refer to Chapter 4, *Creating OLAP Cube Reports*, in this guide.
In order for data to be joined across different data sources in a document, a common attribute is needed across the datasets (see the following illustration)

A common attribute is achieved through mapping objects (attributes and prompts) retrieved from different data sources to existing objects in the MicroStrategy environment. Information on mapping columns for Freeform SQL reports is provided in the online help (search for the “Mapping columns” topic under “Creating Freeform SQL reports”).

For example, in a Report Services document, you have three reports: one standard MicroStrategy report, one Freeform SQL report, and one OLAP cube report using data from SAP BW. All three reports share the same attribute Product. This means that Product is used in the standard report as a project attribute, the Freeform SQL report has one object mapped to Product, and the OLAP cube report also uses Product to map one of its characteristics from the imported SAP BW query cube. As data is joined by Product (the common attribute), the document will be successfully generated.

If each of the three reports originally has a prompt on Product, then the prompt will only be displayed one time; you only need to answer the prompt one time, instead of three times.
Reporting features

This section discusses the following MicroStrategy reporting features in relation to Freeform SQL reports:

- filters
- prompts
- drilling

Filters

A *filter* specifies the conditions that the data must meet to be included in the report results. For information on Filters in general, please refer to Chapter 5, *Filters*, in this guide.

You cannot use existing filters in a Freeform SQL report; however, you can still do filtering by including a `Where` clause in the SQL statement. You can even embed prompt objects in the `Where` clause, if needed. For example,

```
where Year_ID=[Year_prompt]
```

Only two kinds of prompts can be used: value prompts and element list prompts. For more information, please refer to the Prompts subsection in this chapter.

In addition, you can use the view filter functionality for Freeform SQL reports in the same way as for regular reports.

Prompts

A *prompt* is a MicroStrategy object that allows user interaction at report run time. For general information on prompts, please refer to Chapter 9, *Prompts*, in this guide.

For Freeform SQL reports, only two types of prompts are supported—value prompts and element list prompts. To add prompts, you can select from the two options on the Edit menu in the Freeform SQL Editor:
• **Add New Prompt:** launches the Prompt Wizard that allows you to create a new value prompt or an element list prompt.

  ![Lightning bolt]

  **Note the following:**
  
  – Only project attributes can be used to create prompts in Freeform SQL reports.
  
  – Any prompt created this way is saved as a normal object in the metadata.

• **Insert Prompt:** displays the Open dialog box where you can select an existing prompt that you have previously created in the project, either a value prompt or an element list prompt.

  ![Lightning bolt]

  You cannot type the name of an existing prompt directly into the SQL statement.

Once you exit the Prompt Wizard or the Open dialog box, the prompt is inserted automatically into the SQL statement at the current cursor position. If an area in the SQL statement is highlighted, it is replaced by the prompt name. Prompt objects appear in the SQL statement in pink and are enclosed in brackets ([ ]) if the name of the prompt contains any space, for example, where Year_ID = Year_prompt and where Year_ID = [Year prompt].

### Element list prompts

If the prompt is an attribute element list prompt and you use the key word **In**, you need to manually add parentheses around the prompt name in the SQL statement. Note that you can select either a single answer or multiple answers to the prompt, yielding results such as (4) or (1,2,3,4). See the example below.

```sql
select a11.[YEAR_ID] AS YEAR_ID
from [LU_YEAR] a11
where a11.[YEAR_ID] in (Year_prompt)
```
If you use other operators such as =, >, <, or =/, you do not need to add any parentheses around the prompt name, as you do when you use \textbf{in}. However, you can only select a single answer to the prompt. Therefore, make sure that the maximum number of answers allowed for the prompt is set to “1”. See the example below.

\begin{verbatim}
select all.[YEAR_ID] AS YEAR_ID 
from [LU_YEAR] all 
where all.[YEAR_ID] = Year_prompt
\end{verbatim}

\textbf{Value prompts}

Date and number value prompts are properly formatted to the standards of the database platform that the Freeform SQL report is executed against. For example, a date value prompt yields `TO-DATE('08-MAY-74')` for Oracle and “1974-05-08” for DB2.

However, for text value prompts, you need to manually add single quotes (‘’) around the prompt name if you want the text prompt answer to be applied as a text constant. See the example below.

\begin{verbatim}
Select all.[YEAR_ID] AS YEAR_ID 
From [LU_YEAR] all 
Where all.[YEAR_ID] in 'Text_prompt'
\end{verbatim}

You do not need to add the quotes around the prompt name if the answer is part of the SQL command. This feature actually allows you to select objects from multiple few text files that you can use in the SQL statement. See the example below.

\begin{verbatim}
Select Product_ID, Product_DESC, Budget 
From Text_prompt
\end{verbatim}

Adding single quotes or parentheses around the SQL strings of text value prompts or element list prompts, respectively, is a feature with MicroStrategy 8.0.1. If you used the MicroStrategy 8.0 version and upgrade to 8.0.1, you may need to modify your Freeform SQL reports accordingly if they contain these prompts. Otherwise, the reports may fail.
For more information about the value or element list prompts and step-by-step instructions on how to add them into the SQL statement, please refer to the online help (search for “Using Prompts” under the topic “Creating Freeform SQL reports”).

**Optional prompts**

When you create a new prompt to add to the SQL statement, you can make the answer optional by not selecting the “Prompt answer required” option in the Prompt Generation Wizard. Alternatively, if you use an existing prompt, you need to know if this option was selected during the prompt creation (use the Prompt Editor to find out).

No matter whether the prompt is a new or an existing one, if the prompt answer is optional, then make sure that the syntax related to the prompt is also made part of the prompt string. To accomplish that, in the SQL Statement panel highlight the related syntax before and/or after the prompt **together** with the prompt itself, then right-click the highlighted part and select **Prompt-dependent SQL**. The related syntax will turn pink just as the prompt. For example, `where Year_ID in Year_prompt`, instead of `where Year_ID in Year_prompt`.

For step-by-step instructions, please refer to the online help (search for “Adding an optional prompt to the SQL statement”). Taking this step ensures that if the optional prompt is not answered, then neither the syntax nor the prompt will be processed when the report SQL is generated. If you do not take this step, the report will fail.

If you use an older version of MicroStrategy prior to 8.0, make sure that you run the project update for the metadata; otherwise, the optional prompt functionality will not be applied to Freeform SQL reports.
Drilling

Drilling allows you to look at specific data at levels other than what is originally displayed on the grid or graph. For regular reports, you can drill in different directions, such as down, up, or across attributes, based on the system-generated drill maps or custom drill maps. For Freeform SQL reports, support for drilling is limited to only attributes within the Intelligent Cube (see Chapter 2, Reports, of this guide).

This functionality is controlled by the “Drill with Intelligent Cube” privilege.

For example, a report has the Year and Quarter attributes. When you move Quarter off the report and into the Object Browser, you can only drill down from Year to Quarter on the report. However, if both attributes are placed on the report, you cannot drill from either one of them.

If you move Year off the report and into the Object Browser, you cannot drill up from Quarter to Year on the report.
Security for data access

MicroStrategy has a robust security model that enables you to create users and groups, determine how they are authenticated, control what data they can see, and what functional privileges they can have. For detailed information, please refer to the *MicroStrategy System Administration Guide*. This section discusses the Access Control List (ACL) and security filters that relate to Freeform SQL reports only.

Access control list

An access control list (ACL) is a list of users and groups and the access permission that each one has to objects in a MicroStrategy project. Different users may have different permissions on the same object.

When you use existing objects (including project objects) in column mapping, the ACL of these objects remains. However, new objects (attributes and metrics) created in Freeform SQL reports inherit the ACL from what is defined as default in the Project Configuration Editor (*Project Configuration* - > *Project definition* - > *Security* - > *Set Freeform SQL and MDX objects default security*). When you click *Modify*, the Properties [XDA Objects] dialog box is displayed. The Permissions list has the following settings:

<table>
<thead>
<tr>
<th>User</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Full Control</td>
</tr>
<tr>
<td>Everyone</td>
<td>View</td>
</tr>
<tr>
<td>Public/Guest</td>
<td>View</td>
</tr>
</tbody>
</table>

In addition, whoever creates the new objects (attributes, metrics) has the Full Control permission.
In the Properties [XDA Objects] dialog box, you can change the settings of the default ACL for different groups of users. However, note that the changed settings will only affect the new objects (attributes and metrics) created subsequently in Freeform SQL reports, but not those objects created prior to the change.

Security filters

A **security filter** is a filter object that is used to narrow down the result set that users or groups obtain when they execute reports or browse elements. Usually assigned by Administrators, security filters control, at the MicroStrategy level, what warehouse data users can see.

A security filter has three components that are defined by Administrators:

- **Filter expression**: specifies the subset of the data that a user can analyze (for example, Region in (A,B)).
- **Top range attribute**: specifies the highest level of detail that the security filter allows the user to view.
- **Bottom range attribute**: specifies the lowest level of detail that the security filter allows the user to view.

For more information on security filters in general, please refer to Chapter 3, Deploying the system, in the *System Administration Guide*.

As for regular reports in MicroStrategy, security filters can also be applied to Freeform SQL reports. Actually, a security filter for a Freeform SQL report is not much different from that for a standard report. The same concepts still apply.

If you use an older version of MicroStrategy prior to 8.0, make sure that you run the project update for the metadata; otherwise, the security filter functionality will not be applied to Freeform SQL reports.
By default, Freeform SQL reports do not take into account security filters. The Report Designer has to insert a security filter placeholder in a Freeform SQL report and configure it; otherwise, any user can run the report without being limited in the data he or she sees.

Because the SQL statement is static, a security filter string ("where Security Filter" or "and Security Filter") needs to be manually embedded into the statement, such as the following:

```sql
Select Year_ID, Store_ID, M1
From Store_Fact
Where Security Filter
```

The string `Where Security Filter` would be replaced by `Where Store_ID = 1` when the report is executed for a user who has a security filter (`Store@ID = 1`) like the following:

```sql
Select Year_ID, Store_ID, M1
From Store_Fact
Where Store_ID = 1
```

### Parameters for security filters

The following parameters need to be specified when you create security filters in the Freeform SQL Security Filter Editor:

- **Replacement string**: The Replacement String field is located at the bottom of the Freeform SQL Security Filter Editor. The default value for the replacement string is "Security Filter", which is replaced by the security filter condition when the report is generated.

  To complete the string, add "where" or "and" in front of "Security Filter". If there is no valid security filter, then the whole string ("where Security Filter" or "and Security Filter") does not appear in the generated report SQL. For example, following the example mentioned above, when a user without a security filter runs the same report, the SQL looks like the following:
Select Year_ID, Store_ID, M1
From Store_Fact

As you can see, the whole security filter string is dropped from the generated SQL statement.

If you write “where” or “and” directly into the SQL statement in the Freeform SQL Editor, instead of in the Replacement String field in the Freeform SQL Security Filter Editor, the following will happen:

- **For a user with a security filter:** The report will be generated without any problem.
- **For a user without a security filter:** The report will fail due to invalid SQL statement.

- **Attribute mappings:** The Attribute Mapping panel is located on the upper right side of the Freeform SQL Security Filter Editor. This is where you map attribute forms to columns in the database. For every attribute form used in security filter, you need to provide the string that the Engine uses to replace the form itself when building a security filter expression into the SQL statement. This string is also displayed when SQL is generated for the report. For example,

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Form</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>ID</td>
<td>Table1.Cust_ID</td>
</tr>
</tbody>
</table>

- **Ignorable attributes:** specify attributes that may be ignored by the Engine when the report is being generated, even if they appear in a security filter. For example, if you define the following:

- **Ignore:** Customer
- **Security filter definition:** Year = 1998 and Customer = Bob

Only Year = 1998 is applied when the report is generated.
• **Allow security filters with Top and Bottom levels to be evaluated based on the select level of this report:** This check box option is located at the lower part of the Freeform SQL Security Filter Editor. This option means that the **Select Level** of the report (displayed in the line just above this option) is the true level of the data that is to be retrieved for the report when Top and Bottom criteria are evaluated.

   ![Exercise caution with this option:](image)

   - Not selecting this option when the user indeed has Top and Bottom levels defined will cause the report to fail.
   
   - Select this option only when you are sure that your SQL statement does not retrieve data outside the restriction defined by the security filters. This means that you may need to check the security filters for individual users one by one and see how each one may interact with the SQL statement.

**Creating a security filter**

Security filters are created in the Freeform SQL Security Filter Editor, which can be accessed by selecting **Insert Security Filter** option from the Edit menu in the Freeform SQL Editor. For step-by-step instructions, please refer to the online help (search for “Creating security filters for Freeform SQL reports”).

When you close the editor, the security filter string is automatically inserted into the SQL statement at the current cursor location. The string is displayed in an uneditable mode, just like a prompt, and is bold and underlined in green, for example, **Where Store ID = 1**.

You can edit the security filter after it is inserted into the SQL statement by double-clicking it or right-clicking it and selecting **Edit**.
Managed objects

A managed object is just like a normal object except that it is managed by the system and is stored in a special system folder. When you create a Freeform SQL report, you can use existing attributes (including project attributes) to map the columns in the SQL statement in the Freeform SQL Editor. Alternatively, you can create new attributes, attribute forms, and metrics on the fly to which to map the columns in the SQL statement; these new objects are managed objects.

You can find out whether the object (attribute or metric) is a managed object or not by using the Properties dialog box. If it is a managed object, the Location field on the General tab indicates “Managed by the system” (see the graphic below).

For every managed object (attribute or metric) created in Freeform SQL reports, you can perform the following tasks by using the right-mouse click function:
• From the **Search for Objects dialog box**, you can
  – Edit
  – Rename
  – Delete
  – Search for dependents
  – Display in graphical viewer (for logical tables only)
  – Check or redefine properties

• From the **Report Editor**, you can
  – Rename
  – Remove from report
  – Edit
  – Search for dependents
  – Check or redefine properties

• From the **Freeform SQL Editor**, you can
  – Search for dependents
  – Check or redefine properties
In addition, you can view managed objects through the Freeform SQL Editor, where you are defining a Freeform SQL report. The Object Browser displays a Freeform SQL Objects folder (see the following illustration) that contains all the managed objects created for all your Freeform SQL reports, regardless of which database instance is used for which report. These objects do not have mappings outside of the Freeform SQL reports; therefore, they cannot be used in any standard reports; they can only be used in Freeform SQL reports.

As indicated, you can access managed objects by using the Search for Objects function. Make sure you select the **Display Managed Objects** option (in the Search for Objects dialog box, select **Options** from the Tools menu) so the managed objects will be displayed in the search result (see the following graphic).
Once the managed objects are listed in the Search for Objects result (see the following illustration), you can delete, rename, or edit any one of them by right-clicking its name.

Note the following:
When you delete a Freeform SQL report, all the associated managed attributes and metrics used in the report are not deleted automatically. You must use the Search for Objects feature to search for the object first and then manually delete it by right-clicking its name in the search result.

Managed attributes and metrics in Freeform SQL reports are not deleted even if you use the Schedule Administration Tasks -> Delete Unused Managed Objects. Again, you must use the Search for Objects function to make the deletion.

**Editing managed objects**

In the Search for Objects dialog box, you have the option to edit a managed object (attribute or metric) when you right-click its name. The Attribute Editor or Metric Editor is displayed where you can make the modification. In addition, you can also edit any managed attribute or metric directly from the Report Objects window of the Report Editor by taking the following steps:

1. After defining the Freeform SQL report in the Freeform SQL Editor, the report is displayed in the Report Editor.

2. Save the report.

3. In the Report Objects window, right-click the name of the managed object that you want to modify and select Edit.

The Edit option is only available after the report has been saved and has not been changed since the last Save operation.

4. Depending on the object you selected, the same Attribute Editor or Metric Editor as for standard reports is displayed.

5. Edit the object as needed.

When you close the Attribute Editor or Metric Editor, the Report Editor refreshes the object’s definition, which is reflected in the Freeform SQL report when you run it.
Creating Freeform SQL reports

Freeform SQL reports

Freeform SQL reports can be created on Desktop only. However, these reports can be manipulated and executed from both Desktop and the Web. This section describes the process of creating a Freeform SQL report:

- from a database
- from an Excel file
- from a flat file
- using a stored procedure

Creating a Freeform SQL report from a database

The process of creating a Freeform SQL report from a database involves the following general steps:

1. On Desktop, create a database instance for the data source that the Freeform SQL report will run against.

   You must create a database instance before defining a Freeform SQL report.

2. Make the database instance available for Freeform SQL reports (in the Project Configuration Editor, select Project Definition, then Database Instances, and then the database instance name from the Database Instance list).

3. Create a Freeform SQL report by selecting New from the File menu and then Freeform SQL. The Freeform SQL Editor is displayed.

   Access to the Freeform SQL Editor is available only to Desktop Designers with the “Use Freeform SQL Editor” privilege and those with the “Create schema objects” Common Privilege.

4. Below the toolbar, from the database instance drop-down list, select the database instance against which the Freeform SQL report is set to query.
If there is no database instance defined in the metadata, the Freeform SQL Editor cannot be loaded. Instead, the following message is displayed:

“There is no database instance in the metadata. Please create the necessary database instance before using Freeform SQL.”

5  Type your SQL statement in the SQL statement panel.

6  In the Mapping panel, map the columns in the SQL statement to MicroStrategy objects (attribute forms and metrics).

   The number of mappings should match the number of columns in the SQL statement.

7  Insert prompts into the SQL statement, if needed.

8  Insert security filters, if needed.

9  Click **OK** to exit the Freeform SQL Report Editor. The Report Editor is displayed.

10 Define the Freeform SQL report in the same way as you define a standard report, using features such as formatting, sorting, view filters, thresholds, exporting, and so on.

11 Save the Freeform SQL report.

   You must save the report first before you can run it.

12 Run the report.

For more details on the above procedure, please refer to the MicroStrategy online help (search for “Creating Freeform SQL reports”).
Creating a Freeform SQL report from an Excel file

The Freeform SQL reporting feature also allows you to create reports using data from Excel files. The creation process involves the following steps.

Create a table with the Excel file

1. Prepare the Excel file.
   - Make sure that all the columns with data have proper headers: no space in the header name (for example, Category_ID, not Category ID) and are alphanumeric, starting with alphabets (for example, M2004Q1, not 2004Q1M).
   - Make sure that all cells for the ID column have value in them (not empty).

2. Create a table by doing the following:
   - Highlight the rows and columns with the data that you want to create a report with, including the column headers, such as Category_ID and Category_DESC.
     Do not use the column headings (at the top of the Excel spreadsheet, marked as A, B, C...) to select the whole column because doing so may include numerous empty cells with NULL value.
   - Type a name for the highlighted part (rows and columns) in the name box, and then press **ENTER**.
     You may create multiple tables in one Excel file by highlighting different parts of the file and naming them differently.

3. Save the Excel file with a name.
   - Make sure that the file is not password-protected.
Set up the data source (ODBC)

1. From the Control Panel, select **Administrative Tools** and then **Data Sources (ODBC)**. The ODBC Data Source Administrator dialog box is displayed.

2. Select the **System DSN** tab, and then click **Add**. The Create New Data Source dialog box is displayed.

3. Select the **Microsoft Excel Driver** if you are using the Windows platform, and then click **Finish**. The ODBC Excel Setup dialog box is displayed.

4. Enter a **Data Source Name (DSN)** in the space provided.

5. Click **Select Workbook**. The Select Workbook dialog box is displayed.

6. Under Database Name, select the Excel file that you saved and named in Step 1, Prepare the Excel file.

7. Click **OK** to close the Select Workbook dialog box and return to the ODBC Excel Setup dialog box.

8. Click **OK** to return to the ODBC Data Source Administrator dialog box.

9. Click **OK**. The ODBC data source is set up.

You can then use the MicroStrategy ODBC Test Tool to test if data can be retrieved from the table(s) you created from the Excel file.

Make sure to select the correct DSN for the Excel file for the test.

Create a database instance for the Excel file

1. On MicroStrategy Desktop, create a new database instance that points to the DSN for the Excel file.
You could select any database as the Data Connection Type; however, it is recommended that you use Microsoft Access 7.0.

2 Make the new database instance available for your Freeform SQL report.

In the Project Configuration dialog box, select **Project Definition**, then **Database Instances**, and then choose the DSN for the Excel file from the available database instances list.

---

**Create a Freeform SQL report from the Excel file**

1 From the File menu on Desktop or right-click anywhere in the right panel on Desktop, select **New** and then **Report**.

2 In the New Grid dialog box, select **Freeform SQL**.

3 In the Freeform SQL Editor, from the Database Instance drop-down list, select the database instance you created previously (see the “Create a database instance for the Excel file” subsection).

4 In the SQL Statement panel, type in your SQL query.

   Note the following:
   - Column names in the SQL statement have to match the column headers in the Excel file.
   - Case does not have to match, as long as the column names are correct.
   - Do not use the Excel file name as the table name for the “From” clause. Use the table name instead. Remember that the Excel file is the data source that contains the “tables”.

5 In the Mapping panel, map the columns in your SQL statement to attributes and metrics to be used in the MicroStrategy report.
Note the following:

– When mapping the columns, it is important that you follow the same sequence of the columns as they appear in the SQL statement. Doing otherwise will cause the report to fail.

– Make sure that the number of mappings is the same as the number of columns in the SQL statement. For example, if your SQL statement lists 10 columns from which to retrieve data, you should map these columns to exactly 10 objects (including attributes and metrics).

– For each attribute, you must map the ID form at least.

6 Click OK to close the Freeform SQL Editor. The Report Editor opens in Design view by default.

7 Format the Freeform SQL report as you do with a standard report.

8 From the File menu, click Save or Save As.

You must save the report before you can run it. Otherwise, you will see a Desktop message saying that the report “cannot be executed unless it is saved.”

9 In the Save Report As dialog box, enter a name for the report.

10 Run the Freeform SQL report.

Creating a Freeform SQL report from a text file

The Freeform SQL reporting feature also allows you to create reports using data from text files. The creation process involves the following steps:

Prepare the text file for MicroStrategy use

1 Make sure that the file type is text file (.txt).
2. Select a correct delimiter, for example, comma.

3. Make sure that field (column) names appear in the first row of the file and are delimited.

4. Save the text file in a folder, which will be used as the data source for MicroStrategy reports.

---

**Set up the data source (ODBC)**

1. From the Control Panel, select **Administrative Tools** and then **Data Sources (ODBC)**. The ODBC Data Source Administrator dialog box is displayed.

2. Select the **System DSN** tab and then click **Add**. The Create New Data Source dialog box is displayed.

3. Select **DataDirect5.0 Text File** (version 5.00.00.42) as your ODBC driver and then click **Finish**. The ODBC Text Driver Setup dialog box is displayed.

   If you do not have this driver on your machine, you need to install it.

4. On the General tab, enter a **Data Source Name** (DSN).

5. In the **Database Directory** field, provide the path of the directory where you store the text file.

6. Select **Comma** as the Default Table Type.

7. Select the **Column Names in First Line** check box.

8. On the Advanced tab, click **Define**. The Define File dialog box is displayed.

9. Select the text file you want to define and click **Open**. The Define Table dialog box is displayed.

10. In the Table Information section, in the **Table** text box enter the name of the table for the text file, for example, LU_EMPLOYEE (for LU_EMPLOYEE.txt).

11. Select the **Column Names in First Row** check box.
It is important that you select this option.

12 Click **Guess** to display all the columns contained in this table.

13 Click **OK** to return to the Define File dialog box.

14 Click **Cancel** to return to the ODBC Test Driver Setup dialog box.

15 Click **Apply** and then **OK** to return to the ODBC Data Source Administrator dialog box.

16 Click **OK**. Your data source for the text file is now set up.

17 (Optional) Use the MicroStrategy ODBC Test Tool to test retrieving data from the table (for the text file).

*Make sure to select the correct DSN for the text file.*

---

**Create a Freeform SQL report from a text file**

1 From the File menu on Desktop, right-click anywhere in the right panel on Desktop, select **New** and then **Report**.

2 In the New Grid dialog box, select **Freeform SQL**. The Freeform SQL Editor is displayed.

3 Below the toolbar, from the Database Instance drop-down list select the database instance you created in the previous procedure, “Create a database instance for the Excel file”.

4 In the SQL Statement panel, type in your SQL query.  

*Note the following:*

– Column names in the SQL statement have to match the field names in the text file.

– Case does not matter, as long as the column names are correct.
5 In the Mapping panel, map the columns in your SQL statement to attributes and metrics that will be used in the MicroStrategy report.

- Note the following:
  - When mapping the columns, it is important that you follow the same sequence of the columns as they appear in the SQL statement. Doing otherwise will cause the report to fail.
  - Make sure that the number of mappings is the same as the number of columns in the SQL statement. For example, if your SQL statement lists 10 columns from which to retrieve data, you should map them to exactly 10 objects (including attributes and metrics).
  - For each attribute, you must map the ID form at least.

6 Click OK to close the Freeform SQL Editor. The Report Editor opens in Design view by default.

7 From the File menu, click Save or Save As.

- You must save the report before you can run it. Otherwise, you will see a Desktop message saying that the report “cannot be executed unless it is saved”.

8 In the Save Report As dialog box, enter a name for the report and click Save.

9 Run the Freeform SQL report.

Creating a Freeform SQL report using a stored procedure

Creating a Freeform SQL report using a successful stored procedure is similar to creating such a report from a regular database (see “Creating a Freeform SQL report from a database” on page 158). The tricky part is the mapping of columns. Although the stored procedure itself does not display any column names, you need to know in advance what exact columns will be retrieved once the procedure is executed. Otherwise, it may be difficult for you to do the mapping for the columns.
For example, if you use the following stored procedure:

```
Execute sp_customer_profit
```

you may need to map the columns to the following MicroStrategy objects:

- Customer ID
- Customer DESC
- Customer City ID
- Customer City DESC
- Profit

Below are the general steps you need to take when you use a stored procedure to create a Freeform SQL report.

**To use a stored procedure to create a Freeform SQL report**

1. From the File menu on Desktop or right-click anywhere in the right panel on Desktop, select **New** and then **Report**.

2. In the New Grid dialog box, select **Freeform SQL**. The Freeform SQL Editor is displayed.

3. Below the toolbar, from the Database Instance drop-down list, select the database instance you created previously. You can refer to the same procedure as in “Create a database instance for the Excel file”.

4. In the SQL Statement panel, type in your stored procedure.
Different databases use different syntax. Make sure you use the correct one. Below is some information on stored procedure execution for some major databases.

<table>
<thead>
<tr>
<th>Database</th>
<th>Stored procedure</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>call stored_procedure_name with DB2 ODBC drive</td>
<td>The stored procedure must have been created indicating that it has a result set. The results will be sent back to the client.</td>
</tr>
<tr>
<td>Oracle</td>
<td>call stored_procedure_name( ) with MicroStrategy ODBC driver</td>
<td>The stored procedure must return the results into a table that can subsequently be selected.</td>
</tr>
<tr>
<td>SQL Server</td>
<td>exec stored_procedure_name with SQL Server ODBC drive</td>
<td>The stored procedure returns the data to the client. No particular precaution is needed</td>
</tr>
</tbody>
</table>

5. In the Mapping panel, map the columns that are supposed to be retrieved by the stored procedure to attributes and metrics that will be used in the MicroStrategy report.

6. Click **OK** to close the Freeform SQL Editor. The Report Editor opens in Design view by default.

7. From the File menu, click **Save** or **Save As**.
   
   You must save the report before you can run it. Otherwise, you will see a Desktop message indicating that the report “cannot be executed unless it is saved”.

8. In the Save Report As dialog box, enter a name for the report and click **Save**.

9. Run the Freeform SQL report.

For more information, please refer to the MicroStrategy online help.
Introduction

Many companies have both a data warehouse and SAP Business Intelligence Warehouse (SAP BW). This system landscape requires an integrated business intelligence (BI) solution, such as MicroStrategy, that can concurrently access both SAP BW and the data warehouse effectively. This chapter describes how MicroStrategy Intelligence Server integrates with SAP BW by using SAP’s OLAP Business Application Programming Interface (BAPI) and MultiDimensional Expressions (MDX). Specifically, it discusses the following topics:

- MicroStrategy integration with SAP BW (on page 171)
- Understanding the SAP BW terminology (on page 176)
- Relating SAP BW objects to MicroStrategy objects (on page 179)
In addition to concepts related to the MicroStrategy and SAP BW integration, this chapter also provides general information and guidelines on how to perform various types of manipulation in the process of creating OLAP cube reports with SAP BW. For step-by-step instructions, however, you should consult the MicroStrategy online help.

The following SAP Notes provide the latest information on the restrictions and capabilities of SAP's MDX/OLAP BAPI interfaces. Please use your Service Marketplace login to access these notes:

- **820805**—XMLA: No restriction on MEMBER_NAME.
- **820925**—MDX Restrictions.
- **838800**—Lists all the notes that answer questions concerning MDX/OLAP BAPI.
MicroStrategy integration with SAP BW

MicroStrategy provides a rich set of functionality ranging from OLAP Services and Report Services to Narrowcast capabilities, all of which are exposed via a unified Web interface. Using the MicroStrategy standard interface, Intelligence Server can join data from different sources, in addition to relational databases, and bring the data into one single MicroStrategy project. One of the additional data sources is SAP BW.

It is important to understand that the MicroStrategy integration with SAP BW does not change the overall structure of the MicroStrategy product. Rather, MicroStrategy has gained an additional data source for analysis. In other words, SAP BW is simply another data warehouse that holds data for report generation.

SAP BW obtains data from R/3, CRM, SEM, or another SAP data source system. This data is stored in cubes or other SAP objects. To access the data, the Intelligence Server generates MDX. Defined by Microsoft, MDX is similar to SQL but is used to query cubes. An MDX expression returns a multidimensional result set (dataset) that consists of axis data, cell data, and properties data. For more information on the MDX syntax, please refer to http://msdn.microsoft.com/ and search for MDX.

With the SAP BW OLAP BAPI Certification (on MicroStrategy 8.0), MicroStrategy Intelligence Server is certified to connect and execute reports against SAP BW cubes. As SAP’s proprietary API for accessing SAP BW data and functionality, the OLAP BAPI provides an open interface through which Intelligence Server can access the SAP BW data. MicroStrategy has chosen to use the OLAP BAPI approach because it is the most native interface that SAP provides.

With the Powered by Net Weaver Certification (on MicroStrategy 7i -7.5.3), MicroStrategy Web Universal is certified to run on SAP Web Application Server, and MicroStrategy Web and SDK are certified to run with SAP Enterprise Portal through iView Packages.
If you use both SAP BW and MicroStrategy as your BI solution, you can get the best out of both products, including the following:

- access to both SAP and non-SAP data
- five styles of BI
- custom development of reports and applications
- transaction-level analysis
- integration with other systems via Web Services

**Understanding MicroStrategy architecture**

The MicroStrategy platform offers OLAP Services, Report Services, and Narrowcast Server functionality, all of which can be accessed through the Web. Support for SAP BW and Freeform SQL provides additional mechanisms for pulling data into this platform for analysis, as illustrated in the following diagram.

For information on Freeform SQL reporting, please refer to Chapter 3, *Creating Freeform SQL Reports*, in this guide.
Data is pulled from multiple SAP BW systems using MDX and operational systems using Freeform SQL. Once the data is retrieved, it is treated in the same manner as data pulled from the relational data warehouse. This means that core MicroStrategy capabilities are available no matter what the original data source is.

To understand the current MicroStrategy architecture better, let us look at the basic object model of MicroStrategy 7i and how various data sources were accessed then.
Object model in MicroStrategy 7i

In the 7i metadata model (see illustration below), you could have multiple MicroStrategy projects, each pointing to a data warehouse, which was represented by the database instance. One database instance could be referenced by multiple projects in a configuration. Each project contained one schema object that held the logical model for that project. When a report was executed, the SQL Engine would implicitly reference the schema to determine which table(s) should be queried.

As illustrated, a Report Services document in the 7i model may contain multiple datasets, but the only source is the data warehouse.

Object model in MicroStrategy 8

In the MicroStrategy 8 model (see the following illustration), a project is extended to access the SAP BW data through a separate database instance. However, note that instead of pointing to the schema object, each OLAP cube report points directly to one cube in MicroStrategy, which is a logical placeholder for a physical cube that exists in SAP BW. Each report can only reference one specific cube, due to the
structure in SAP BW where queries can only be run against one cube at a time. However, note that you can create multiple reports to run against one cube, and that a single MicroStrategy project may reference multiple database instances, each of which represents a distinct OLAP cube provider.

Also in this model, you can include any number of standard reports, Freeform SQL reports, and OLAP cube reports in one Report Services document. By bringing these different types of reports together inside a document, report designers can create rich reports and analytics that take advantage of data from both data warehouses and SAP BW. For information on Report Services documents, please refer to the MicroStrategy Document Creation Guide. For information on Freeform SQL reports, please refer to Chapter 3, Creating Freeform SQL Reports, in this guide.
Understanding the SAP BW terminology

Before looking in depth into the MicroStrategy integration with SAP BW, you need to be familiar with the terms that are used to describe the SAP BW objects. Some of these terms are provided in the following. Further information is provided later in this chapter on how the SAP BW objects are related to those in the MicroStrategy environment (see Relating SAP BW objects to MicroStrategy objects).

For comprehensive and detailed explanation on SAP BW objects, please refer to documentation by SAP.

- **InfoObject**: is the master data, such as characteristics and key figures, which are equivalent to attributes and facts in a MicroStrategy project.

- **InfoProvider**: is a generic term defining all SAP BW data structures available for reporting and analysis purposes such as the following:
  - **InfoCube**: is a multidimensional cube (see more below).
  - **ODS object**: is an operational data store object. ODS objects are flat relational tables and are similar to MicroStrategy fact tables.
  - **MultiProvider**: a logical union of two or more InfoProviders that are used to combine data from two different subject areas, for example, three InfoCubes or two ODS objects.

- **InfoCube**: is the primary object that SAP BW uses to store data for analysis. InfoCubes define a specific domain of analysis in special areas, for example, finance or sales. Data is organized by dimension and stored physically in a star schema. The fact table at the center of an InfoCube contains the data available for analysis.

- **Query cube (or query)**: defines a subset of data from an InfoCube or another InfoProvider. A query cube includes characteristics (attributes) and key figures (metrics) from its source provider. The relationship between the InfoCube and the query cube is very similar to how a MicroStrategy report includes a subset of modeled attributes and metrics that are available in the data.
Query cubes generally offer better performance than InfoCubes because they are smaller and can be scheduled and cached within SAP BW. Query cubes also provide MicroStrategy users access to additional InfoProviders including ODS objects, InfoSets, and MultiProviders.

Any existing query can be released for analysis within MicroStrategy. All you need to do is to select the **Allow External Access to This Query** check box under the Extended tab in the SAP Query Properties dialog box in the Query Analyzer interface. With this option enabled, designers can quickly access existing query cubes and Business Content when working in MicroStrategy.

- **Characteristic**: provides classification possibilities for a dataset, such as sales region, product, customer group, fiscal year, and period. For example, characteristic Sales Region can have North, Central, and South specifications.

SAP BW characteristics are similar to MicroStrategy attributes. However, when each characteristic is translated into a cube, it is treated as a separate dimension for analysis. In addition, hierarchies can be associated with a specific characteristic within SAP BW. These hierarchies are also available when you work with a cube in MicroStrategy.

- **Key figure**: describes numeric data, such as revenue, profit, and number of call centers. There are five types of key figures: amount, quantities, numbers, date, and time, all of which can be used in InfoCubes, ODS objects, and master data attributes. You can also create calculated key figures and restricted key figures in the query definition in the Business Explorer. This is similar to creating derived metrics and conditional metrics within the MicroStrategy environment.
• **Hierarchy**: is a way of defining the relationships among elements within a characteristic. For example, the characteristic Item might have a hierarchy that includes Category, Subcategory, and finally Item. This is a different paradigm from MicroStrategy’s model where each attribute defines its own level. However, as noted later, when the levels of a hierarchy are viewed in MicroStrategy, they are presented with the traditional attribute-based parent-child relationships.

• **Variable**: is used as a parameter of a query in SAP BW. Defined in the Query Designer, variables can be of such types as characteristic values, hierarchies, hierarchy nodes, texts and formula. When the query is being executed, these variables are filled with values by the system or by the user.

When an OLAP cube is imported into a MicroStrategy project, all the variables in this cube are represented as prompts. When the OLAP cube is used to create a MicroStrategy report, the report inherits all those prompts. In addition, standard prompts can also be created for this report. More information on variables is provided later in this chapter.

When working in MicroStrategy, you will find a list of available cubes for reporting, including all of the published query cubes, InfoCubes, and MultiProviders. For step-by-step instructions on how to create MicroStrategy reports out of the data from SAP BW cubes, please refer to the MicroStrategy online help.

Besides the above-mentioned terminology, you also need to have a basic understanding of the SAP Query Designer, where you define queries. You can select and combine InfoObjects or reusable structures for an InfoProvider and specify the view of the data (query view) by distributing them to filters, rows, columns, and free characteristics. For more information, please refer to documentation provided by SAP BW.
Relating SAP BW objects to MicroStrategy objects

As a Web or Desktop Analyst, you can treat SAP BW reports just as if they were standard MicroStrategy reports. However, if you are a Report Designer, it is helpful to understand how SAP’s metadata model is translated into MicroStrategy’s metadata model.

The translation process involves the following steps:

- **First, from SAP BW to ODBO**: SAP exposes its query cubes and InfoCubes to Intelligence Server through the ODBO model. ODBO stands for OLE DB for OLAP and is a protocol defined by Microsoft. ODBO defines an object model that is used in conjunction with MDX to query cubes. The ODBO model is similar to SAP’s standard model, but not identical. Thus, when thinking about SAP objects, keep in mind how those objects appear in ODBO.

- **Second, from ODBO to MicroStrategy**: After SAP objects are translated into the ODBO model, they are then translated into the MicroStrategy metadata model. You can then interact with SAP content while working within the paradigm that is consistent with the rest of MicroStrategy’s products.
The following illustration demonstrates how the SAP BW objects are exposed in ODBO and then how they are related to objects in the MicroStrategy environment.

The following subsections (each starting with a table) describe each level of comparison from top to bottom as shown in the above illustration.

<table>
<thead>
<tr>
<th>SAP BW</th>
<th>ODBO</th>
<th>MicroStrategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>InfoCube</td>
<td>catalog</td>
<td>(catalog)</td>
</tr>
</tbody>
</table>

- **SAP BW**: InfoCube

Each InfoCube that has queries associated with it is exposed as a catalog in ODBO. Query cubes are accessed through their respective InfoCube catalogs.

- **ODBO**: catalog

Catalogs are used to group cubes. Therefore, ODBO catalogs are exposed in a few editors when selecting and managing cubes.

- **MicroStrategy**: (catalog)

Each catalog includes one InfoCube and associated query cubes, if any. Catalogs in MicroStrategy are represented in a folder.
**SAP BW** is not supported.

**ODBO**: schema

Schema in ODBO provides another grouping mechanism.

**MicroStrategy** is not supported.

<table>
<thead>
<tr>
<th>SAP BW ---</th>
<th>ODBO ---</th>
<th>MicroStrategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>schema</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SAP BW ---</th>
<th>ODBO ---</th>
<th>MicroStrategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>InfoCube/query cube</td>
<td>cube</td>
<td>cube</td>
</tr>
</tbody>
</table>

**SAP BW**: InfoCube/query cube

**ODBO**: cube

**MicroStrategy**: cube

A MicroStrategy cube is an object that is used to map the levels of an SAP BW cube into the MicroStrategy environment. Cubes are treated in a manner very similar to tables in the MicroStrategy metadata. In the same way that a regular table maps the physical columns of a relational table to attributes and metrics, a MicroStrategy cube maps the physical columns of an SAP BW cube to attributes and metrics. The cube may be used to represent InfoCubes, MultiProviders, and query cubes.

<table>
<thead>
<tr>
<th>SAP BW ---</th>
<th>ODBO ---</th>
<th>MicroStrategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>characteristic</td>
<td>dimension</td>
<td>dimension</td>
</tr>
</tbody>
</table>

**SAP BW**: characteristic
Characteristics in SAP BW are similar to attributes in MicroStrategy. For example, an InfoCube might include the characteristic Month, which represents months just like it does in MicroStrategy.

A characteristic appears as a dimension for MicroStrategy users. Each characteristic (dimension) has at least one hierarchy with two levels: the first level is an aggregate of all the data, and the second level is the detailed data. A characteristic may have any number of additional hierarchies, each with \( n \) levels. These hierarchies appear as hierarchies to MicroStrategy users.

For example, you can build a Time hierarchy that is attached to the characteristic Month. This hierarchy defines a number of levels including Year, Quarter, and Month. However, these same levels could either be specially defined as part of the hierarchy, or they could be other characteristics that are used to define the levels of this one hierarchy. For more information, see the following subsection.

Dimensions in SAP BW are used to group characteristics and are not exposed through the ODBO interface. They can only be seen inside the SAP BEx Query Designer when you build a query cube.

Shared dimensions allow a designer to use only one definition for a dimension across multiple cubes. Each characteristic in SAP is modeled as a dimension in ODBO and is shared across cubes. Therefore, all dimensions in cubes coming from SAP BW are shared.

- **ODBO**: dimension

  A dimension in ODBO defines a logical category of analysis. For example, Time and Geography are dimensions along which one might slice data.

  Measures (metrics) are stored in a special measure dimension. In this way, measures are simply one more dimension of a cube.

  Measures in ODBO are called key figures in SAP BW, which are very similar to metrics in MicroStrategy, and they are represented as physical columns.
• **MicroStrategy**: dimension

A dimension object in MicroStrategy is very similar to an ODBO dimension. It is used to group attributes and define parent-child relationships.

<table>
<thead>
<tr>
<th>SAP BW --&gt;</th>
<th>ODBO --&gt;</th>
<th>MicroStrategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>hierarchy</td>
<td>hierarchy</td>
<td>hierarchy</td>
</tr>
</tbody>
</table>

• **SAP BW**: hierarchy

• **ODBO**: hierarchy

• **MicroStrategy**: hierarchy

Hierarchies are used to group attributes (levels) together and define the relationships between these attributes. MicroStrategy reuses the hierarchy objects to represent both dimensions and hierarchies from ODBO.

<table>
<thead>
<tr>
<th>SAP BW --&gt;</th>
<th>ODBO --&gt;</th>
<th>MicroStrategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual level</td>
<td>level</td>
<td>attribute</td>
</tr>
</tbody>
</table>

• **SAP BW**: virtual level

Levels are generated automatically based on either the definition of the characteristic or the hierarchies associated with a characteristic.

SAP BW levels have names such as Region Level 01, Region Level 02, and so on. The inclusion of the term “Level” is an SAP BW convention. In MicroStrategy, architects have the option to rename the levels of a cube with a more readable convention.

• **ODBO**: level
• **MicroStrategy**: attribute (ID/DESC)

MicroStrategy attributes map to ODBO levels. However, each ODBO level generates two physical columns and forms in MicroStrategy—ID and DESC.

<table>
<thead>
<tr>
<th>SAP BW ---&gt;</th>
<th>ODBO ---&gt;</th>
<th>MicroStrategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>characteristic value</td>
<td>member</td>
<td>attribute element</td>
</tr>
</tbody>
</table>

• **SAP BW**: characteristic value

• **ODBO**: member

• **MicroStrategy**: attribute element

Element values come from either the database or a cube. For example, 1998 and 1999 are elements of the attribute Year.

<table>
<thead>
<tr>
<th>SAP BW ---&gt;</th>
<th>ODBO ---&gt;</th>
<th>MicroStrategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>characteristic attribute</td>
<td>property</td>
<td>attribute form</td>
</tr>
</tbody>
</table>

• **SAP BW**: characteristic attribute

• **ODBO**: property

• **MicroStrategy**: attribute form

Attribute forms provide additional information about a given attribute. For example, the attribute Customer may have the forms First Name and Last Name. This concept also applies to ODBO and SAP BW.

 explodes

In SAP BW, forms are sometimes referred to directly as attributes. SAP BW also supports navigational attributes. These attributes are presented as distinct dimensions when working in MicroStrategy.
SAP BW variables

Variables are used in SAP BW to parameterize queries. There are several types of variables, including characteristic values, hierarchies, hierarchy nodes, texts and formula elements. When the query is being executed, these variables are filled with values. For detailed information on variables, please refer to documentation by SAP.

Variable types with the Replacement Path, Customer Exit/SAP Exit and Authorization processing types are automatically resolved by the SAP BW system. Only variables with the Manual Entry/Default processing type are presented to users for resolution.

Originally created in an SAP query cube, variables are represented as prompts in the MicroStrategy environment. The conversion process involves the following general steps:

1. When an SAP query cube is imported into a MicroStrategy project, variables are automatically turned into prompts in the MicroStrategy OLAP cube.

2. When a MicroStrategy report is created using the MicroStrategy OLAP cube, the report inherits the prompts included in this cube.

   On top of the “inherited” variable prompts, additional standard MicroStrategy prompts can also be created for the report. For more information, please see the Prompts section on page 199 in this chapter.

Mapping between variables and prompts can be viewed in the OLAP Cube Catalog, which lists all the prompts that were converted from variables, in addition to cube names, dimensions, and measures/key figures (see the following graphic).
In the OLAP Cube Catalog, you can view any variable’s properties by right-clicking its name and then selecting **Properties**. Details about this variable in SAP BW are displayed on the Variable tab, including Variable Name, Variable Type, Selection Type, Entry Type, Default Low, Default Low Description, and Variable Ordinal.

In addition, using the right-mouse click you can also **Edit** the prompt in the Prompt Generation Wizard, **Rename** the prompt, or **Map** the variable to a prompt in an existing MicroStrategy project.
One prompt can be mapped to more than one variable across cubes. When executing a Report Services document with multiple datasets using these cubes, you see this prompt displayed only once. This allows the same prompt answer to be used to resolve multiple variables during document execution.

After an OLAP cube report is executed, you can view the prompt details in the Report Details pane in the Report Editor. This is especially useful if you want to get a summary of the variable elements that are used in answering the variable prompts. For more information, please see Prompts on page 199.

The following table contains information on how the different types of SAP BW variables are mapped to MicroStrategy prompts.

<table>
<thead>
<tr>
<th>SAP Variable Type</th>
<th>MicroStrategy Prompt</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristic Value variable</td>
<td>Element list prompt or attribute qualification prompt</td>
<td>See the note below for more information.</td>
</tr>
<tr>
<td>Hierarchy variable</td>
<td>N/A</td>
<td>Not supported.</td>
</tr>
<tr>
<td>Hierarchy Node variable</td>
<td>Hierarchy element list prompt</td>
<td>Both single and multiple selection are supported.</td>
</tr>
<tr>
<td>Text variable</td>
<td>N/A</td>
<td>Not available from SAP BW.</td>
</tr>
<tr>
<td>Formula variable</td>
<td>Value prompt: all types</td>
<td>No major changes.</td>
</tr>
</tbody>
</table>

Characteristic value variables offer an “Including/Excluding” option. Qualifications in the Including section cause the data to be brought into the query, while those in the Excluding section restrict the data from being displayed in the query. To be consistent with the SAP functionality, the MicroStrategy interface qualifies on the key value of each element by default.

If you use any SAP BW key date variables in your query, you need to manually set the variable’s property in the OLAP Cube Catalog, so it is distinguished from a simple characteristic variable on date:

1. Right-click the variable name and select Properties. The Properties [variable name] dialog box is displayed.
2 On the Variable tab, select the **Set Key Date** check box. And then click **OK**.

**SAP BW structures**

Structures in an SAP BW query cube define the two axes of a query (rows and columns) and are of two types: key figure structures and characteristic structures.

Each element of a key figure structure is represented as a unique metric in the MicroStrategy environment. In addition to key figure structures, a query could also have characteristic structures, each of which is represented as a single flat dimension with one level. This representation is consistent with how characteristic variables are represented in SAP BW through the OLAP business application programming interface (BAPI).

In a MicroStrategy report, you cannot drill down into the elements of characteristic structures.

**Using the OLAP Cube Catalog**

Once you understand the relationships among the objects in SAP BW, ODBO, and MicroStrategy, you can start working with the SAP BW data in MicroStrategy. The best place to start with is the OLAP Cube Catalog, where you can both import cubes and remap the cubes before you create any OLAP cube reports. Just as the Warehouse Catalog, the OLAP Cube Catalog can be accessed from the Schema menu on Desktop.

The OLAP Cube Catalog option is available only after an SAP BW database instance has been created.

This section discusses how you can use the OLAP Cube Catalog to bring the SAP data into a MicroStrategy project and what functions you can perform once the data is brought in. For step-by-step instructions on how to use the OLAP Cube Catalog, please refer to the online help (search for “Using the OLAP Cube Catalog”).
Importing cubes

SAP BW cubes can only be imported into a MicroStrategy project by an Architect with the **Import OLAP Cubes** privilege.

Cube importing is performed on the Cube Selection tab (see the following illustration). When you open the OLAP Cube Catalog, all the SAP BW cubes (InfoCubes and associated query cubes, if any) are displayed, by default, under their respective catalog names in the Available Cubes pane. Using the plus (+) or minus (-) sign next to a catalog name, you can expand or hide the cubes contained in this catalog. A catalog is marked with an icon showing a folder containing a cube, an InfoCube is marked with a cube icon in blue, and a query cube is marked with a cube icon in green.

Select **Find** from the Edit menu or click the Find icon on the toolbar to open the Find dialog box to search for a specific cube that you want to use for your report.
You can import InfoCubes and query cubes by using the > arrow to move the selected cubes from the Available Cubes pane to the Selected Cubes pane. For step-by-step instructions, please refer to the MicroStrategy online help (search for “Using the OLAP Cube Catalog”).

If you change your mind about the selected cubes, you can right-click any InfoCube or query cube in the Selected Cubes pane, and select **Remove [cube name]**. Using the right-mouse click, you can also select the **Update Structure** option to synchronize with the updated definition of cube structures in SAP BW, for example, when a new characteristic or key figure has been added to the InfoCube in SAP BW.

### Using the Select Cube dialog box

When you create an OLAP cube report, you have to go through the Select Cube dialog box, where you select cubes for your report. This dialog box can also be used by an Architect with the “Import OLAP cubes” privilege to import cubes by using the “Retrieve cubes” option, which is available only after an SAP database instance has been defined. For detailed information, please refer to the online help (search for “Using the Select Cube dialog box”).
You can click the Find button at the bottom of this dialog box to open the Find dialog box, where you can search for a specific cube for your report by the cube’s name. For details, please refer to the online help topic “Finding cubes”.

**Mapping cubes**

When an OLAP cube is imported into MicroStrategy, the Intelligence Server creates new objects (attributes, metrics, and hierarchies) that reflect the levels of the imported cube (see the following illustration). Using the Cube Mapping feature in the OLAP Cube Catalog, you can map attribute forms for each attribute contained in the imported cube (by default, only the ID and DESC forms have been automatically mapped). In addition, you can remap the attributes and metrics to existing objects in a MicroStrategy project. This capability allows data to be joined across sources in Report Services documents, thus maintaining a consistent logical model.
As shown in the illustration, the Physical View in the left pane represents the cube structure in SAP BW. The characteristic (dimension) is located at the very top (with a green chart and box symbol), hierarchy is below the dimension (with a green stacked boxes symbol), and attribute is represented by a green symbol with two side-by-side rectangles. The Logical View in the right pane represents the equivalent structure in MicroStrategy, with the same symbols for hierarchies and attributes as in standard reports.

In the Physical View pane, you can use the plus (+) sign next to the attribute levels to display the attribute forms. By default, only the ID and DESC forms are displayed. Use the **Display All properties** icon on the toolbar to show additional attribute forms and then do the mapping for each one of them. For each attribute, the ID form must be mapped at least. You can also use the **Show Technical Names** icon on the toolbar to display the SAP BW terms for each attribute and its attribute forms.

For attributes (characteristics) and metrics (key figures), you can perform the following manipulations by right-clicking the name in either the Physical View pane or the Logical View pane:

- **Edit** the attribute or metric in the Attribute Editor or the Metric Editor that is displayed.
- **Rename** the attribute or metric so it has a different name in the MicroStrategy project from the name of the characteristic or key figure that it is mapped to in SAP BW.
- **Map** the characteristic or key figure to an existing attribute or metric in the MicroStrategy project.
- **Check the Properties** of the characteristic or key figure. In the Properties dialog box, you can view the information on its Name, Technical Name, and Description in SAP BW.

For variables, you can also perform the above-mentioned manipulations. However, note that when you select **Edit**, the Prompt Generation Wizard is displayed. This is because variables in SAP BW are represented as prompts in MicroStrategy. For more information, please refer to *SAP BW variables* on page 185.
By default, all the hierarchies are set to Balanced. However, if you know that the structure of a hierarchy is unbalanced or ragged, take the following steps to change its properties:

1. In the OLAP Cube Catalog Editor, right-click the hierarchy name in the Physical View pane. The Properties dialog box is displayed.

2. On the Hierarchies tab, select the check box This hierarchy is unbalanced or ragged and then click OK. The word “(Unbalanced)” will be displayed next to the name of the hierarchy in the Logical View pane.

Failure to set the hierarchy correctly may cause wrong data to be retrieved for the OLAP cube report.

For step-by-step instructions on mapping and remapping objects from SAP BW to MicroStrategy objects, please refer to the MicroStrategy online help (search for the “Mapping cubes” topic).

How SAP BW objects are mapped to MicroStrategy objects

In MicroStrategy 7i, when an Architect was defining a project, much of that process would center on identifying logical entities, such as attributes and facts, that existed in physical tables. For example, an Architect might identify that the key for the attribute Customer existed in the table LU_CUSTOMER. The Architect would then define a logical and physical model in the MicroStrategy metadata. This model was what the SQL Engine would reference to generate SQL at run time.

In the context of SAP BW, an OLAP cube, instead of a single table, contains all the metadata information necessary to define a logical model and physical model. When the Architect needs to add a cube to a project in MicroStrategy, he or she simply needs to select a cube by using the OLAP Cube Catalog or Select Cube dialog box, as described previously.
By default, a MicroStrategy cube is created that maps to the definition of the source cube in SAP BW and uses attributes and metrics, which are used only within this particular cube’s environment. Although these objects are new and are part of the project, they do not relate to the project’s schema. A new schema is created for each SAP BW database instance used in a MicroStrategy project. Once an SAP BW cube has been imported, it can be used to build reports and documents in MicroStrategy.

Why do you need to remap the cubes?

Although you can use the auto-generated attributes and metrics, you may want to remap them to existing objects in the MicroStrategy project for the following reasons:

- Remapping allows a Report Designer to integrate the logical model of the project with the data in the cube that has been imported.

- Remapping facilitates joining data across sources within a Report Services document.

  For example, if an OLAP cube-based report and a standard report both use the attribute Year, then Year can be used to group the data within a document.

- Remapping allows an administrator to search for dependents and manage access control lists (ACLs) more easily.

Note that while the levels of a cube can be remapped by an Architect, the nature of the cube is not changed. Remapping simply replaces the attributes or metrics that are used to represent the cube’s structure with attributes or metrics in an existing MicroStrategy project. In the case of attributes, they can be remapped to project attributes that participate in the ROLAP schema. In the case of metrics, they can only be remapped to ones that exist in the SAP BW environment. For example, three cubes can share the same metric “Revenue”. 
Example 1: Unmapped cube

The following diagram shows two logical models. The one on the left exists in a specific cube, and the one on the right in a MicroStrategy project. Although both models have a Time hierarchy, none of the individual attributes are shared.
**Example 2: Partially mapped cube**

Example 2 (see the following diagram) also shows two logical models. The difference between the two examples is that the cube has been partially remapped so that it shares the attributes Year, Quarter, and Month.

The dimensions of SAP BW cubes are always shared. Therefore, when a level is remapped, that change will apply to all the cubes that share that dimension. In this case, changes to the Time dimension would likely apply to most cubes in the project that contains this dimension.

**When should you remap cubes?**

Although you can remap the columns either when a cube is first imported or later on after you have created a project, it is recommended that you do the remapping initially so that subsequent users can take advantage of it. This also prevents maintenance issues because reports can need to be modified if a cube is remapped after report creation. Finally, remapping the levels of a cube facilitates joining data within a Report Services document through the use of the Group By feature.
Reporting features

This section discusses the following reporting features:

- filters
- prompts
- drilling
- setting display hierarchy

Filters

For OLAP cube reports using the SAP BW data, most of the filtering features remain the same as those for standard MicroStrategy reports. For information on filters in general, please refer to Chapter 2, Filters, in this guide and the online help.

In a standard report, the filter is evaluated independently of the template in most cases. However, in an OLAP Cube report, due to the nature of MDX, there is a close relationship between the objects in the filter and the objects in the Report Objects list (base template). In an OLAP Cube report, qualifications on dimensions that do not participate in the template are evaluated as a filter prior to metric aggregation. Qualifications on dimensions that participate in the template are applied as a limit after metric aggregation.

For example, if Year is on the template and Year is qualified on, then it will restrict the rows of data without changing any of the individual metric values. However, if Year is qualified on and Store is on the template, then each metric value will be restricted to considering only those years determined by the filter.

Metric qualifications that have a specific output level are evaluated along with that attribute, either before or after aggregation, but a metric limit (a qualification at the report level) is always applied as a limit.
The logical relationships between filters are set automatically depending on the dimension(s) to which the filtered objects belong. The following rules govern the logical operators between qualifications, due to the nature of the structure of the MDX query language:

- Filters that define attributes in the same dimension are joined by the **Or** operator.

  For example, Customer and Customer Region both belong to the Customer dimension. Therefore, you could have filters as follows:
  
  - Customer In list (Addison Don, Adess Merrell, Adler Keith)
  - OR
  - Customer Region In list (France, Germany)

- Filters that define attributes in different dimensions are joined by the **And** operator.

  For example, Category belongs to the Product dimension, and Year belongs to the Time dimension. Therefore, you could have filters as follows:
  
  - Category In list (Electronics, Movies)
  - AND

- Metric limits are always joined by the **And** operator with all the other filters, such as follows:

  - Quarter In list (2003 Q2, 2004 Q2)
  - AND
  - Evaluate the Set at Report level where (Revenue Greater than 40 )
  - AND
  - Evaluate the Set at Report level where (Revenue Less than 100 )

No matter whether the objects are on or not on the report template, you create filters for them in the same Report Filter pane in the Design view. While defining the filters, you can even build prompts into the filters. For step-by-step instructions on how to create a filter for an OLAP cube report, please refer to the online help.

- SAP BW does not support string operators. Therefore, qualifications such as like, contains, and so on cannot be processed.
Note that attribute qualifications on the DESC attribute form are not supported, unless the DESC property has been remapped to one of the extended properties of the level in the OLAP Cube Catalog.

In addition, if you need to import a text file for attribute element qualification, note that the same rules apply to OLAP cube reports as to standard reports. Namely, data in the file needs to be one of the following:

- tab-delimited
- return-delimited
- list-delimited as specified in the Regional Settings

**Prompts**

As discussed in the *SAP BW variables* section, MicroStrategy turns the variables to prompts when SAP BW cubes are imported into a project. In addition to these inherited prompts, you can create standard prompts in the same way as you do in standard MicroStrategy reports.

You can choose to display prompt details for OLAP cube reports just as for standard reports by opening the Desktop Preferences dialog box, selecting **Reports, Show report details**, and then **Include prompt details**. This feature is especially useful if you want to get a summary of the variable elements that are used in answering the variable prompts.
There are two ways you can use to create a standard prompt in an OLAP Cube report:

- **Converting report filters to prompts**: After you create a report filter for the OLAP Cube report, you can convert the filter to a prompt in the Report Editor by right-clicking the report filter name and selecting *Convert to Prompt* (see the following illustration).

- **Prompt Generation Wizard**: This procedure is the same as in a standard MicroStrategy report. You can create all kinds of prompts depending on your needs. For more information, please see the online help (search for the “Creating prompts in OLAP cube reports” topic).

  Use the attributes and metrics to qualify prompts by browsing to the *Data Explorer* folder.
When you save a prompted report, whether with inherited prompts converted from SAP BW variables or standard prompts, you have the option to save it as a static or prompted report (see the following illustration).

Drilling

For OLAP cube reports, standard drilling is supported within the cube. This means that you can drill up or down within the dimension that the attribute belongs to as well as in other directions to dimensions included in the same cube. Drill paths are automatically generated based on the definition of the cube.

When you run an OLAP cube report on the Web, it is recommended that you use the sub-menu display option (select Preferences, then Project Defaults, then Drill Mode, and then select Display advanced drill options as sub menus on the context menu).
Setting display hierarchy

When you create an OLAP cube report in the Report Editor, the Object Browser displays one hierarchy for each dimension in a cube that you selected. Only one hierarchy is displayed because SAP BW supports only one hierarchy in a query. If a dimension has multiple hierarchies, SAP BW assigns a default display hierarchy for it; and MicroStrategy inherits this default setting when the OLAP cube is imported.

To change the default hierarchy display, right-click the hierarchy in the Object Browser window in the Report Editor and select **Set Display Hierarchy**. For more information, please see the online help.

Note the following:

- The Display Hierarchy option is not available if there is only one hierarchy in a dimension.

- If objects of the currently displayed hierarchy are used on the template or referenced in any filter on the report, then when you select the **Set Hierarchy Display** option, the following message is displayed:

  “This hierarchy is currently used in this report. Please remove any references to this hierarchy before changing the display hierarchy.”

Related features

This section discusses the following MicroStrategy features in relation to OLAP cube reports:

- managed objects
- authentication
Managed objects

When an OLAP cube is imported into a project, managed objects (attributes, metrics, columns, tables, and so on) are created to describe that cube. A managed object is just like a normal object except that it is managed by the system and is stored in a special system folder that is invisible to users. However, one way to access managed objects is by using the Search for Objects function from the Tools menu on Desktop.

In the Search for Objects dialog box, select the Display Managed Objects option (from the Tools menu, select Options) so they will be displayed in the search result. Once the managed objects are listed in the search result, you can rename or edit any of them by right-clicking its name.

A managed object can be removed once it is no longer referenced by another unmanaged object in the project. The removal of unused managed objects is usually performed by the Administrator. For more information, please refer to the MicroStrategy System Administration Guide.

Authentication

Most of the features of authentication that apply to standard MicroStrategy reports also apply to OLAP cube reports, described as follows:

- **Standard authentication, LDAP authentication, NT Authentication**: are supported independent of the data source that is being used, for example, relational databases, OLAP cube providers.

- **Connection mapping**: is supported in the same way as for standard MicroStrategy reports. In addition, specific connection maps may be designated for each database instance, user/group combination.

- **Warehouse pass-through authentication**: is supported in the same way as for relational data providers. If multiple sources are configured for warehouse pass-through execution, then the same login information must be applicable to all sources.
To enforce SAP BW security in MicroStrategy, it is recommended that you use connection mapping or pass-through authentication.

For information on authentication in general, please refer to the *MicroStrategy System Administration Guide* and the online help.

## Connecting to SAP BW servers

This section discusses how to connect to SAP BW servers in the Windows and the Unix environment. For more information, please refer to the online help (search for “Connecting to SAP BW”).

### Windows

**Important note from SAP:**

Starting with JCo 2.1.4 and JCo 2.0.11, you are required to install the new Visual Studio .NET C/C++ run-time libraries on Windows platforms. Please see [SAP Note 684106](https://service.sap.com/~form/sapnet?_SHORTKEY=01100035870000463649) for details on how to install them.

Take the following steps to connect to SAP BW servers in Windows:

1. Open the **SAP Service Marketplace** and download the **SAP Java Connector**.

**Note the following:**

- The SAP Java Connector can be downloaded from SAP Service Marketplace: `service.sap.com` -> support -> download or using this link: https://service.sap.com/~form/sapnet?_SHORTKEY=01100035870000463649. Use your own logins to access the SAP Service Marketplace.

- MicroStrategy certifies version 2.1.3 and supports more recent versions.
2 Install the **Java Connector**.

3 Place the following files in a directory that is indicated in the Path environment variable:
   - `Librfc32.dll`
   - `Sapjcorfc.dll`

4 Place the `Sapjco.jar` in the Common Files MicroStrategy folder, for example, `C:\Program files\Common files\MicroStrategy`.

5 Create a database instance that points to SAP BW by selecting SAP BW as the Warehouse Type and providing the following information as required:
   - **Application Server**: name of the SAP BW Server or IP address
   - **SAP Router String**, if you use an SAP Router
   - **System Number** from the SAP BW system
   - **Client Number** from the SAP BW system
   - **Language**: the language code provided by your SAP administrator, for example, `EN` for English.

   Note the following:
   - You may get the above information from your SAP Logon.
   - Create a database login with the user and password that you want to use to connect to SAP BW.

**UNIX**

Take the following steps to connect to SAP BW servers in Unix:

1 Open the **SAP Service Marketplace** and download the **SAP Java Connector**.
Note the following:

- The SAP Java Connector can be downloaded from SAP Service Marketplace: service.sap.com -> support -> download or using this link: https://service.sap.com/~form/sapnet?_SHORTKEY=01100035870000463649. Use your own logins to access the SAP Service Marketplace.

- MicroStrategy certifies version 2.1.3 and supports more recent versions.

2 Select the zip file for the platform you want to use and unzip it, for example, gunzip or gzip [file name].

3 Search for the files listed in the following table, copy them onto your machine, and create a new directory for them. For example,

- for AIX: /home/dsmith/SAP/AIX
- for SUN: /home/dsmith/SAP/SUN

<table>
<thead>
<tr>
<th>AIX</th>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>librfccm.o</td>
<td>librfccm.so</td>
</tr>
<tr>
<td>libsapcorfc.so</td>
<td>libsapcorfc.so</td>
</tr>
<tr>
<td>sapjco.jar</td>
<td>sapjco.jar</td>
</tr>
</tbody>
</table>

4 Edit the SAP.sh file in the installation directory.../env/SAP.sh by doing the following:

1) Add the Write and Execute privileges to this file. The default is Read Only.

You can type in the command “chmod+wx file name” in the Unix console.

2) Open the SAP.sh file and enter the information for XXXX_SAP_LIBRARY_PATH=’’. This information indicates where the server needs to point in order to use the downloaded files.
For example, for AIX:

```
# set up the environment for SAP

XXXX_SAP_LIBRARY_PATH="/home/dsmith/SAP/AIX/

if [ -n "${XXXX_SAP_LIBRARY_PATH}" ];
    then xxxx_append_path LD_LIBRARY_PATH
    "${XXXX_SAP_LIBRARY_PATH:?}"
    export LD_LIBRARY_PATH
fi
```

For example, for SUN:

```
# set up the environment for SAP

XXXX_SAP_LIBRARY_PATH="/home/dsmith/SAP/SUN/

if [ -n "${XXXX_SAP_LIBRARY_PATH}" ];
    then xxxx_append_path LIBPATH
    "${XXXX_SAP_LIBRARY_PATH:?}"
    export LIBPATH
fi
```

3) Save the file.

5 Add sapjco.jar to the installation directory: /install/jar.  

   Make sure that you have the Write privilege to this directory.

6 Restart the server to get the latest updates.
7 Create a database instance pointing to the SAP BW server of your choice, providing the following information, as required:

- **Application Server**: name of the SAP BW Server or IP address
- **SAP Router String**, if you use an SAP Router
- **System Number** from the SAP BW system
- **Client Number** from the SAP BW system
- **Language**: the language code provided by your SAP administrator, for example, EN for English.

You can use either the Database Instances Editor or the Database Instance Wizard to create a database instance for SAP BW. For more information, please refer to the online help.

**Creating OLAP cube reports**

OLAP cube reports can be created on Desktop only. Once created, they can be manipulated and executed from the Web as well.

To create an OLAP cube report, you need to be a Desktop Designer with the “Define OLAP cube report” privilege.

Take the following steps to create an OLAP cube report:

1. On Desktop, select **New** from the File menu and then select **Report**.

   Alternatively, right-click in an empty space in the right panel on Desktop and select **Report**.

2. In the New Grid dialog box, select **OLAP Cube Report** and then click **OK**. The Select Cube dialog box is displayed.

3. Select a database instance from the drop-down list for Select OLAP Server.
4 If cubes have already been imported by an Architect (which is normally the case), they are displayed in their respective catalog structure (a catalog contains an InfoCube and query cubes, if any). If this is the case, click the plus sign (+) next to the catalog name to display its content (InfoCube and query cubes), and then select the cube that you want to use for your report.

You can use the Find dialog box to search for a specific cube that you want to use for your report.

If you are an Architect with the “Import OLAP Cubes” privilege and if no cubes have been imported, then use the “Retrieve Cubes” option (disabled for those without the privilege) at the bottom of the dialog box to import the cubes from SAP BW.

You can also use the OLAP Cube Catalog to import cubes, as described in the Importing cubes section on page 189.

5 (Optional) Select the Display technical names check box to show the SAP BW technical names for the cubes.

6 Click OK to open the Report Editor.

7 Select attributes and metrics from the Object Browser and put them on the grid.

8 Create a filter, if needed.

9 Run the report.

For more information on the above procedure, refer to the MicroStrategy online help (search for the “Creating OLAP cube reports” topic).
Filters

Introduction

A filter specifies the conditions that the data must meet to be included in the report results. It is an easy way to answer both simple and complicated business questions.

An example of a simple business question is the sales for all stores in the Northeast. To build this report, place the attribute Store and the metric Dollar Sales on your report and filter on Northeast. The filter allows you to view the results for only those stores in the Northeast.

You are interested in reviewing the sales for only those stores that have at least one category with a margin greater than 20% and total sales of more than $100,000 for the year. This is a more complicated question, but it can also be answered using a filter.

This chapter reviews the categories of filter functionality and the types of filtering techniques used, to help you achieve the answers to your simple and complex business questions.

Remember, all that a filter really does is help you answer “show me X where Y is true.”

This chapter does not include information about security filters, which are discussed in the MicroStrategy System Administration Guide.
Types of filters

In the reporting environment, when you design a report, it queries the database against all the data stored in the data warehouse. By applying a filter, you can narrow the data to consider only the information that is relevant to answer your business question. The previous chapter, Reports, discussed the following three ways to restrict data to generate reports:

- A report filter that is a criterion used to select the data to calculate the metrics in the report.
- A report limit that specifies a set of criteria used to restrict the data returned in the report data set after the report metrics are calculated. Because it is based on the report’s metric values, a limit is applied after all metrics are calculated.
- A view filter that is an additional filter applied in memory to the report data set. It affects only the view definition.

This chapter focuses on report filters and expands on their advanced capabilities. For more information and examples of the other methods, refer to Chapter 2, Reports.

Report filter options

The following options are available while creating filters:

- Attribute qualification allows you to filter on an attribute’s form (ID, description, and so on) or on elements of the attribute.
- Set qualification allows you to create a set based on one of the following:
  - a metric (also known as a metric qualification)
  - a relationship filter
- Shortcut to a report, which is also referred to as Report as filter, uses an existing report as a filter.
- Shortcut to a filter uses an existing filter as a base to which you can add more conditions to further refine the filter.
• Advanced qualification lets you create one of the following:
  – a custom expression, including a relationship filter
  – a joint element list, which allows you to join attribute elements and then filter on that result set

You can also create filters that prompt you for answers when you run the report. These prompted filters allow the report to have dynamic report definitions, which can change with each query by altering the information in the prompt dialog box. All of these options are available in the Filter Editor, and are discussed in detail in this chapter.

Attribute qualification

*Attribute qualifiers* enable you to specify conditions that attribute elements must satisfy to be included in the filter definition. For example, you can create a qualification on the attribute Month so that the result set returns only months beginning with the letter “J.”

Attribute element list qualification

Attribute element list qualifications allow you to qualify on a list of attribute elements. For example, in a report, you can use an attribute element list qualification on the attribute Customer, to return data only for those customers that you specify in your list.
## Attribute element list qualification example

This example refers to filters and reports saved in the MicroStrategy Tutorial. The directory path within Desktop is Public Objects\Reports\Technical Reports\Reports by Feature\Advanced Reporting Examples. You can follow the steps to interact with the filters and reports, or you can view the samples without creating your own. Remember to save any objects that you create under a different name, so that you do not overwrite the samples in the MicroStrategy Tutorial.

A report includes the revenue, cost, and profit for all employees. However, certain months are not representative of the normal business cycle, so they should be excluded from the report calculations. To do that, create a filter that excludes the months of April, May, and December. This filter is saved as Month in the Supporting Objects subdirectory. For step-by-step directions on creating a filter, see the online help.

Open the Basic Report. Note that Leanne Sawyer’s contribution to revenue is $316,786. Now switch to Design View and add the Month filter. When you re-execute the report, it looks like the following.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$354,350</td>
<td>$269,591</td>
<td>$84,759</td>
</tr>
<tr>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$215,628</td>
<td>$163,478</td>
<td>$52,150</td>
</tr>
<tr>
<td></td>
<td>Kieferson</td>
<td>Jack</td>
<td>$229,233</td>
<td>$173,672</td>
<td>$55,561</td>
</tr>
<tr>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td>$198,976</td>
<td>$150,803</td>
<td>$48,173</td>
</tr>
<tr>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$286,039</td>
<td>$215,651</td>
<td>$69,388</td>
</tr>
<tr>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$220,460</td>
<td>$167,350</td>
<td>$53,110</td>
</tr>
</tbody>
</table>

If you do not want to create it yourself, this report is saved as Filter - Month Filter in the Tutorial.

Notice that the metrics have different values than in the Basic Report. Sawyer’s contribution to revenue is now $198,976. In the Basic Report, all data for all months was retrieved from the data warehouse. The Revenue metric was calculated using all months. In this filtered report, April, May, and December amounts are not retrieved from the data warehouse, so the metric cannot include them in its calculations.
Attribute form qualification

Attribute form qualifications allow you to qualify on an attribute form. For example, to return data only for those customers whose last names start with the letter H, you can use an attribute form qualification for the form Customer Last Name in a report.

Attribute form qualification example

This example refers to filters and reports saved in the MicroStrategy Tutorial. The directory path within Desktop is Public Objects\Reports\Technical Reports\Reports by Feature\Advanced Reporting Examples. You can follow the steps to interact with the filters and reports, or you can view the samples without creating your own. Remember to save any objects you create under a different name, so that you do not overwrite the samples in the MicroStrategy Tutorial.

A report includes the revenue, cost, and profit for all employees. You want to view the data of only those employees whose last name begins with the alphabet ‘B’. To do this, create a filter that qualifies on the Last Name of the attribute Employee. Choose the Operator as Begins With and Value as B. Save this filter. For step-by-step directions on creating a filter, see the online help.

Open the Basic Report. Now switch to Design View and add this filter. When you re-execute the report, it looks like the following.

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mid-Atlantic</td>
<td>Bernstein</td>
<td>Lawrence</td>
<td>$403,122</td>
<td>$305,790</td>
<td>$97,322</td>
</tr>
<tr>
<td></td>
<td>Brown</td>
<td>Vernon</td>
<td>$548,862</td>
<td>$416,009</td>
<td>$132,853</td>
</tr>
<tr>
<td>Southeast</td>
<td>Benner</td>
<td>Ian</td>
<td>$526,867</td>
<td>$399,590</td>
<td>$127,277</td>
</tr>
<tr>
<td>Northwest</td>
<td>Becker</td>
<td>Kyle</td>
<td>$692,441</td>
<td>$525,800</td>
<td>$166,641</td>
</tr>
<tr>
<td>Southwest</td>
<td>Bates</td>
<td>Michael</td>
<td>$733,886</td>
<td>$594,787</td>
<td>$139,099</td>
</tr>
<tr>
<td></td>
<td>Bell</td>
<td>Catlin</td>
<td>$624,961</td>
<td>$473,529</td>
<td>$151,432</td>
</tr>
</tbody>
</table>
Notice that the report displays the revenue of only those employees whose last name begins with the alphabet ‘B’.

**Dynamic dates**

When you qualify on a date attribute form with the date data type, you can select dynamic dates, which are fixed offsets of the current date. They can be either a fixed set of dates or different date ranges that change through time. For example, a dynamic date can be used in a report that examines sales in the previous two months. This would be represented as "today" with an offset of two months. You can express Dynamic date qualifications in several ways, as shown in the following examples:

- an offset of four years, three months, two weeks, and one day from today
- Monday of this week
- Monday of this week with an offset of two days
- the fourth of this month
- the fourth Wednesday of this month
- May fourth of next year
- the third Wednesday in May of this year

While evaluating a dynamic date such as “first of this month minus seven days,” the order in which these two parts are calculated is significant. The addition or subtraction of days, weeks, months, or years is always done first, before “first of this month,” “this week,” “this year,” and so on is calculated. For example:

- If today is February 13th, then “today minus seven days” is February sixth, and “the first of the month of today minus seven days” is February first.

- However, if today is February second, then “today minus seven days” is January 26th, and “the first of the month of today minus seven days” is January first.
**Imported filter**

You can import filter elements into the Filter Editor from sources other than MicroStrategy, if you choose the attribute qualifying operator as **In list** or **Not in list**. To import elements into a filter, the elements should be stored in an Excel file or a text file.

The import filter elements option adds more flexibility to the Filter Editor by allowing lists of data from pre-existing files to be imported into the filter definition. Existing filter definitions can also be exported to a file.

You can use a prompt to allow you to select the file to import when you run the report.

Importing elements from a text file or a Microsoft Excel file can be quicker and more efficient than selecting each individual element to be included in the filter. For example, you have an Excel spreadsheet that lists the products on sale this month. You need to review last week’s revenue for just these items. Rather than selecting them in the Filter Editor, you can simply import the file. Likewise, you can export existing filter definitions to a file.

The following rules apply to the formatting of files:

**Excel**-Data can be stored in rows, columns, or both.

1. If the data in a cell has double quotes in the first and last position, it is imported as it is, with the quotes.

2. If the data in a cell has single quotes in the first and last position, it is imported as is, with the quotes.

3. If the data in a cell does not satisfy conditions 1 or 2, it is checked to see if it is a number. If the data is a number, it is imported as it is.

4. If the data in a cell does not does not satisfy conditions 1 or 2, it is checked to see if it is a date. If it is a date, it is imported by adding single quotes at the beginning and at the end to comply with the date format.
If the data in a cell does not satisfy any of the above conditions, it is considered as text data and is imported by adding double quotes at the beginning and end to comply with the text format.

Text - Data in a text file must be one of the following:

- Tab-delimited
- List-delimited as specified in the regional settings
- Return-delimited

Attribute-to-attribute qualification

Attribute-to-attribute qualifications allow you to create reports that compare two attributes through attribute forms. For example, using attribute-to-attribute qualifications, by comparing order date with ship date, you can create a report that displays the orders that were shipped within a week of their order date.

Attribute-to-attribute qualification example

This example refers to information found in the MicroStrategy Tutorial.

An attribute-to-attribute qualification can be used to create a report that lists orders that were shipped more than 27 days after the order date. Start a new report with Order, Day, Ship Date, Revenue, Cost, and Profit. To limit the amount of data considered for the report, add a filter for December 2003. Finally, create the attribute-to-attribute qualification as outlined below.
To create an attribute-to-attribute qualification

1. Double-click in the Report Filter pane to create a new qualification.

2. Select **Add an Attribute qualification** and click **OK**. The Attribute Qualification dialog box opens.

3. Find the attribute **Ship Date** in the Object Browser (in the Customer hierarchy) and drag it to **Attribute** in the Attribute Qualification dialog box.

4. Change **Qualify on** to **ID**.

5. Change the **Operator** to **Greater than**.

6. Select **Custom** and enter the following:

   \[(\text{Day@ID} + 27)\]

   This adds 27 days to the Day attribute, which is the order date. The Ship Date is compared to this value.

7. Click **OK**.

Execute the report, which displays as shown below.

<table>
<thead>
<tr>
<th>Order</th>
<th>Day</th>
<th>Ship Date</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>39025</td>
<td>12/31/2002</td>
<td>1/28/2003</td>
<td></td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>39030</td>
<td>12/6/2002</td>
<td>1/3/2003</td>
<td></td>
<td>14</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>39054</td>
<td>12/30/2002</td>
<td>1/27/2003</td>
<td></td>
<td>190</td>
<td>139</td>
<td>51</td>
</tr>
<tr>
<td>39112</td>
<td>12/27/2002</td>
<td>1/24/2003</td>
<td></td>
<td>15</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>39132</td>
<td>12/30/2002</td>
<td>1/27/2003</td>
<td></td>
<td>13</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>39143</td>
<td>12/5/2002</td>
<td>1/6/2003</td>
<td></td>
<td>74</td>
<td>56</td>
<td>18</td>
</tr>
<tr>
<td>39236</td>
<td>12/31/2002</td>
<td>1/28/2003</td>
<td></td>
<td>13</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>39249</td>
<td>12/29/2002</td>
<td>1/26/2003</td>
<td></td>
<td>38</td>
<td>29</td>
<td>9</td>
</tr>
<tr>
<td>39320</td>
<td>12/12/2002</td>
<td>1/9/2003</td>
<td></td>
<td>199</td>
<td>148</td>
<td>51</td>
</tr>
</tbody>
</table>

This report is saved as **Attribute to Attribute Comparison**.
The first order, Order 39025, was ordered on 12/31/2002 and shipped on 1/28/2003. That is a difference of 28 days.

Attribute Qualification Prompt

An attribute qualification prompt allows you to qualify on the values of attribute elements, attribute forms, or operators when you run a report. You can create the following types of attribute qualification prompts:

- **Choose from all attributes in a hierarchy** allows you to choose an attribute to qualify on when you run a report. You are, however, restricted to choosing just the attributes from the selected hierarchy. After selecting the attribute, you can qualify on the ID or create an element list filter.

- **Choose from an attribute element list** allows you to apply qualifications to an attribute form. You can choose an attribute from a list of attributes and qualify on the elements of the attribute.

- **Value prompt** allows you to select a single value on which to qualify, such as a date, a specific number, or a specific text string.

- **Qualify on an attribute** allows you to apply qualifications to an attribute form. You can choose an attribute from a list of attributes and qualify on an attribute form.

Set qualification

A set qualification allows you to create a set based on either a metric qualification or a relationship qualification.
Set qualification: metric qualification

Metric qualifiers enable you to restrict metric values based on value, rank, or rank percentage. Metric qualifiers restrict the amount of data used to calculate the metrics on a report. For example, a store manager might want to see sales numbers for products whose current inventory levels fall below a certain level. This report will not necessarily display the inventory figures for those products.

Output level

The output level specifies the level at which the metric is calculated for the qualification. For example, if the metric qualification is Sales > 1000, Sales could mean sales per day, month, year, store, region, and so on. Creating a set qualification with an output level of store is equivalent to having a fixed list of stores, if you knew which ones met the metric qualification, in a simple attribute qualification. However, the list of stores in the qualification is generated dynamically.

The output level can be specified in several ways.

- An attribute list allows you to specify the exact set of attributes (such as day, month, or year) to use as the output level.

- Report level means that the output level is defined by the level of the report that contains the metric qualification. For example, if the lowest level of the report is year and the output level is set to report level, the metric is calculated for the year.

- Metric level means that the output level is defined by the level, or dimensionality, of the metric itself, regardless of the level of the report.

- The None selected option calculates the results at the report level if any of the following is true:
  - The metric is a compound metric.
  - The metric’s dimensionality is set to report level.
  - The metric’s dimensionality is set to nothing.
Otherwise, the metric’s dimensionality is used.

If you do not select an output level, the None selected option is used by default.

**Break by**

This advanced function of a metric qualification allows you to choose the attribute level at which to restart counting rank or percent values for a metric. This level must be greater than or equal to the level of aggregation for the metric itself, as shown in the following example.

Given the following data:

<table>
<thead>
<tr>
<th>Region</th>
<th>Market</th>
<th>Store</th>
<th>Actual ($K) Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Mid-Atlantic</td>
<td>Baltimore</td>
<td>40</td>
</tr>
<tr>
<td>Northeast</td>
<td>Mid-Atlantic</td>
<td>Philadelphia</td>
<td>30</td>
</tr>
<tr>
<td>Northeast</td>
<td>New England</td>
<td>Boston</td>
<td>20</td>
</tr>
<tr>
<td>Northeast</td>
<td>New England</td>
<td>Greenwich</td>
<td>10</td>
</tr>
</tbody>
</table>

If you specify “Break by Market,” the ranking counter is reset for each market (in descending order).

<table>
<thead>
<tr>
<th>Region</th>
<th>Market</th>
<th>Store</th>
<th>Actual ($K) Sales</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Mid-Atlantic</td>
<td>Baltimore</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>Northeast</td>
<td>Mid-Atlantic</td>
<td>Philadelphia</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>Northeast</td>
<td>New England</td>
<td>Boston</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>Northeast</td>
<td>New England</td>
<td>Greenwich</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>
If you specify “Break By Region,” the ranking counter is reset for each region (in this example, as there is only one region, the counter is not reset).

<table>
<thead>
<tr>
<th>Region</th>
<th>Market</th>
<th>Store</th>
<th>Actual ($K) Sales</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Mid-Atlantic</td>
<td>Baltimore</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>Northeast</td>
<td>Mid-Atlantic</td>
<td>Philadelphia</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>Northeast</td>
<td>New England</td>
<td>Boston</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>Northeast</td>
<td>New England</td>
<td>Greenwich</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>

**Merge attribute qualifications**

The **Advanced** button allows you to specify whether existing attribute qualifications should be merged into the calculation of the metric qualification. By default, this option is selected, combining the qualifications.

A metric qualification is contained in a separate pass of SQL, creating a temporary table or “mini-report.” If the qualifications are merged, attribute qualifications are added to that pass of SQL. If they are not merged, the attribute qualifications are not included in the metric qualification. They instead appear in the main SQL pass.

For more information on how metric qualifications work, see *An alternative explanation of metric qualification* in Chapter 2, *Reports.*
For example, a report shows revenue by region. The report filter contains the attribute qualification of year equal to 2002 and the metric qualification of revenue over $1 million. If the default is kept, the qualifications are merged. Only 2002 revenue is considered when the metric checks for revenue over $1 million. The report results are:

<table>
<thead>
<tr>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$ 1,230,989</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>$ 1,784,622</td>
<td></td>
</tr>
<tr>
<td>Southeast</td>
<td>$ 1,044,998</td>
<td></td>
</tr>
<tr>
<td>Southwest</td>
<td>$ 1,475,579</td>
<td></td>
</tr>
</tbody>
</table>

In contrast, if the qualifications are not merged, revenue is calculated for all time before the metric qualification is evaluated. However, only revenue from the year 2002 is displayed on the report. As shown in the following sample, regions are included that do not have $1 million of revenue in 2002, but do have $1 million of revenue across time.

<table>
<thead>
<tr>
<th>Region</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>$ 1,230,989</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>$ 1,784,622</td>
<td></td>
</tr>
<tr>
<td>Southeast</td>
<td>$ 1,044,998</td>
<td></td>
</tr>
<tr>
<td>Central</td>
<td>$ 942,718</td>
<td></td>
</tr>
<tr>
<td>South</td>
<td>$ 727,680</td>
<td></td>
</tr>
<tr>
<td>Northwest</td>
<td>$ 800,487</td>
<td></td>
</tr>
<tr>
<td>Southwest</td>
<td>$ 1,475,579</td>
<td></td>
</tr>
<tr>
<td>Web</td>
<td>$ 869,557</td>
<td></td>
</tr>
</tbody>
</table>

Besides affecting the report results, merging the qualifications reduces the amount of data a calculation must process.
Metric-to-metric comparison

Metric-to-metric comparisons allow you to create reports that dynamically compare the values of two metrics. For example, you can create a report that restricts the data to revenue greater than last quarter’s revenue.

Create a report that displays the revenue for Call centers Atlanta, San Diego, and Miami for each quarter during the year 2002. To do this, create one filter that includes the call centers Atlanta, San Diego and Miami and another filter that includes the year 2002. For step-by-step directions on creating a filter, see the online help.

When you execute the report, it looks like the following:

<table>
<thead>
<tr>
<th>Call Center</th>
<th>Quarter</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>Q1 02</td>
<td></td>
<td>$131,604</td>
</tr>
<tr>
<td></td>
<td>Q2 02</td>
<td></td>
<td>$121,566</td>
</tr>
<tr>
<td></td>
<td>Q3 02</td>
<td></td>
<td>$120,400</td>
</tr>
<tr>
<td></td>
<td>Q4 02</td>
<td></td>
<td>$195,356</td>
</tr>
<tr>
<td>San Diego</td>
<td>Q1 02</td>
<td></td>
<td>$288,073</td>
</tr>
<tr>
<td></td>
<td>Q2 02</td>
<td></td>
<td>$341,784</td>
</tr>
<tr>
<td></td>
<td>Q3 02</td>
<td></td>
<td>$229,975</td>
</tr>
<tr>
<td></td>
<td>Q4 02</td>
<td></td>
<td>$395,128</td>
</tr>
<tr>
<td>Miami</td>
<td>Q1 02</td>
<td></td>
<td>$91,743</td>
</tr>
<tr>
<td></td>
<td>Q2 02</td>
<td></td>
<td>$117,008</td>
</tr>
<tr>
<td></td>
<td>Q3 02</td>
<td></td>
<td>$86,216</td>
</tr>
<tr>
<td></td>
<td>Q4 02</td>
<td></td>
<td>$181,105</td>
</tr>
</tbody>
</table>

Now, create a revenue metric that calculates the revenue for the previous quarter and save it as **RevenueLastQuarter** metric. Create a metric-to-metric comparison filter that uses the Revenue metric. Choose the Function as **Metric Value**, and Operator as **Greater than**. Choose the Value as **Metric** and browse to select the newly created metric **RevenueLastQuarter**. Save the filter **LastQuarter**. The report, when re-executed with the **LastQuarter** filter, now looks like the following:
Note that only those revenues whose values are greater than the revenues of the previous quarter are displayed on the report.

### Set qualification: relationship qualification

*Relationship qualification* allows you to create a link between two attributes and place a filter on that relationship. It allows you to create a set of elements from an attribute based on its relationship with another attribute. For example, relationship filtering allows you to create a report that shows you all the stores selling Nike shoes in the Washington, DC area or all customers who have checking accounts but no saving accounts.

You can create relationship filters using either Set qualification or Advanced qualification in the Filter Editor. Set qualification provides an interface to guide you through the process, while you must enter commands in Advanced Qualification. The syntax for the Advanced qualification is described in *Advanced qualification: relationship filters*.

You have the following options while creating a relationship qualification:

- **Output level** is the level at which the set is calculated. You can select the attribute(s) for the output level.

- **Filter Qualification** defines the input filtering criteria, that is, the relationship on which to qualify. You can select an existing filter or create a new filter.
• **Relate output level and filter qualification by** is the relation between the attributes in the output level and filter qualification. The relation can be a fact, a table, or an empty filter. If the relationship is left empty, the schema is used to select the appropriate table.

For example, to create a report that shows all the stores selling Nike shoes in the Washington, DC area, you need to set the output level to Stores, the filter qualification to Nike shoes and Region, and the relation to the fact Sales.

**Metric qualification prompt**

A metric qualification prompt allows you to select a function, or an operator, or specify the value for a metric, when you run a report. You can create the following types of metric qualification prompts:

• **Qualify on a metric prompt** allows you to qualify on a metric. You can choose a metric by specifying a single metric to use when the report is run or by specifying a search object to restrict the list of metrics from which you can choose a metric, when a report is run.

• **Metric Object prompt** allows you to select one or more metrics that meet specific criteria when a report is run. For example, you could use a search to return a list of all metrics that contain a certain fact. From that list of metrics, you can then choose the metrics that you want to see on the report.

• **Value prompt** allows you to select a single value on which to qualify, such as a date, a specific number, or a specific text string.

**Shortcut to a report**

A shortcut to a report qualification is also known as *Report as filter*. In the Desktop, you select **Add a Shortcut to a Report** to access the *report as filter* functionality.
The report data set of an existing report can be used as a filter for another report. Often, the result of one report is exactly what is needed as a filter in another report. Rather than create a filter that mimics the results of a report, that report itself can be used as a filter in the second report. When used as a filter, only the report’s data definition is considered; any changes to the view definition do not influence the filter conditions.

Using reports as filters provides a more visual way of building reports and analyzing their results. It also provides a fluid transition from viewing data in a report to analyzing additional reports based on the data in the original. Report as filter is a different way to achieve the same results as a metric qualification, but it is easier to understand and create.

Reports with consolidations or custom groups cannot be used as a shortcut to a filter.

An example of a report used as a filter is included in Report as filter example in Chapter 2, Reports.

**Report Object Prompt**

The report object prompt allows you to choose the results of one report to be included in another report. You can define a report object prompt by specifying a search object or specifying a predefined list of objects to choose from, while executing a report.

**Shortcut to a filter**

Creating a shortcut to a filter allows you to use an existing filter, or add conditions to that filter, to apply to a report. In generic terms, Filter1 contains two conditions, A and B. You can use Filter1 in another filter and add another condition, C, to it. The data must then satisfy all three conditions - A, B, and C - to be included.
For example, you are a manager in New England, responsible for stores in Boston, Providence, and Greenwich. Your project contains a filter called Stores in my Region, which is defined as the Boston, Providence, and Greenwich stores. The Women’s Clothing filter includes the classes Blouses and Dresses. A third filter, All Days in Dec 01, is a date range that includes all the days in the month of December, 2003.

To study December sales in your stores for women’s clothing, create a new filter. Include a shortcut to each of the three filters.

**Filter Object Prompt**

The filter object prompt allows you to choose the filters to be included in a report. You can define a filter object prompt by specifying a search object or specifying a predefined list of objects to choose from, while executing a report.

**Advanced qualification: custom expression**

Advanced qualifications allow you to create custom expressions to fit particular needs. For example, you can create a relationship filter using the custom expression area of the advanced qualification window.

**Advanced qualification: relationship filters**

The advanced qualification window allows you to use commands rather than an interface. To work with an interface, see *Set qualification: relationship qualification*.

The following syntax is required to create a relationship filter using an advanced qualification:

```
<relation; (filter qualification)>
{list of output attributes}
```

where:
• The relation can be a fact, a table, or an empty filter. Facts and tables are relationships between attributes in Filtering Input and Output Level. Relationships determine which table is used during SQL generation.

If a relationship is left empty, the schema is used to select the appropriate table.

• The filter qualification defines input filtering criteria. It may consist of an attribute qualification, a filter qualification, or a metric qualification, followed by a comma and an output level.

• The list of output attributes is a comma-separated list of the attributes to be filtered on. If your regional settings are not set to English, the list separator must be whatever is defined in your regional settings. The output level dictates the contents of the relationship filter output set.

It is easiest to simply drag an attribute from the Object Browser into the list. If you manually enter the attribute, it must be in the format

- `[attributename]@ID` or
- `[attributename]@DESC`.

For example, if you are creating a report that shows all stores selling Nike shoes in the DC area, the relationship filter syntax looks like this:

```
<Fact Sales>; [Nike Shoes, Region]>
{Stores@ID}
```

where Fact Sales is the table name, Nike Shoes and Region form the filter qualification, and Stores is the attribute.

**Advanced qualification: apply functions**

Pass-through expressions, or apply functions, in MicroStrategy are intended to provide access to the special functions or syntactic constructs that are not standard in MicroStrategy, but are found in various RDBMS platforms. There are five predefined apply functions, each belonging to a different function type:

- **ApplySimple**—Single-value function
• ApplyAgg—Group-value function
• ApplyOLAP—OLAP function
• ApplyComparison—Comparison function
• ApplyLogical—Logical function

For more information, refer to the MicroStrategy Analytical Functions Reference.

Among these five functions, ApplyComparison and ApplyLogical can be used to create custom expressions for filtering.

While an Apply function can be used wherever the function group it belongs to is applied, you should NOT use any Apply functions when standard MicroStrategy functions can be used to achieve the goal. This is because using Apply functions effectively bypasses the validations and other benefits of the product. Therefore, use it ONLY when support does not exist in the MicroStrategy product and submit an enhancement request so that MicroStrategy can evaluate your needs for inclusion in a future product release.

For details on the apply functions, see Appendix C, Pass-through Expressions.

**Advanced qualification: joint element list**

*Joint element lists* allow you to choose attribute elements from different attributes to filter the report result set. Unlike attribute qualifications, joint element lists also allow you to join attribute elements and then filter on that attribute result set. In other words, you can select specific element combinations, such as quarter and category. As in the report sample included below, you can filter on electronics in Q1 2003 and music in Q3 2003.
Joint element list example

This example refers to information saved in the MicroStrategy Tutorial.

Before creating a joint element list, you must ensure that the Advanced Qualification option is displayed on the Filter Editor. From the Desktop, complete the following steps:

1. Select My Preferences from the Tools menu.
2. Choose the Editors tab.
3. Click Filter Options.
4. Select Show advanced qualification, if it is not already selected.
5. Click OK to return to the Desktop.

Open the Basic Report. Note that Leanne Sawyer’s revenue is $316,786. This is sales for all time and all categories. You need to see revenue for specific quarter and category combinations, for example, electronics in Q1 2003 and music in Q3 2003. To do this, switch to Design View and create a joint element list, as described below.

To create a joint element list

1. Double-click in the Report Filter pane to add a new qualification.
2. Select Add an Advanced Qualification and click OK. The Advanced Qualification pane opens.
3. Select Joint Element List from the Option pull-down list.
4. Select Category and Quarter from the Available attributes list and click > to add them to the Selected attributes list.
5. Click the Add icon to the right of the Element list. The first value in each attribute is added to the list.
6 Click the **Modify** icon to the right of the Element list. The Select Element List dialog box opens.

7 Double-click **Electronics** to change the category.

8 Select **Quarter** from the Available Elements drop-down list.

9 Double-click **Q1 03** to change the Quarter.

10 Click **OK** to return to the Advanced Qualifications dialog box.

11 Click the **Add** icon to add another element. Again, the first value in each attribute is added by default.

12 Select the new element and then repeat steps 6 through 10, this time changing Category to **Music** and Quarter to **Q3 03**.

13 Click **OK** to save the new qualification.

Execute the report. The results are displayed below:

<table>
<thead>
<tr>
<th>Region</th>
<th>Employee</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>De Le Torre</td>
<td>Sandra</td>
<td>$33,842</td>
<td>$25,951</td>
<td>$7,891</td>
</tr>
<tr>
<td></td>
<td>Kelly</td>
<td>Laura</td>
<td>$16,767</td>
<td>$12,921</td>
<td>$3,846</td>
</tr>
<tr>
<td></td>
<td>Kieffer</td>
<td>Jack</td>
<td>$31,292</td>
<td>$23,594</td>
<td>$7,708</td>
</tr>
<tr>
<td></td>
<td>Sawyer</td>
<td>Leanne</td>
<td><strong>$18,901</strong></td>
<td>$14,498</td>
<td>$4,413</td>
</tr>
<tr>
<td></td>
<td>Sonder</td>
<td>Melanie</td>
<td>$28,152</td>
<td>$21,435</td>
<td>$6,667</td>
</tr>
<tr>
<td></td>
<td>Yager</td>
<td>Beth</td>
<td>$21,608</td>
<td>$16,613</td>
<td>$4,995</td>
</tr>
</tbody>
</table>

This report is saved as **Joint Element List**.

Notice that Sawyer’s revenue is now only $18,901. The decreased revenue reflects the qualification, since only sales for electronics in the first quarter of 2003 and the sales for music in the third quarter of 2003 are included in the metric calculations.
Introduction

Metrics are MicroStrategy objects that represent business measures and key performance indicators. They represent calculations to be performed on data stored in the database, and are similar to formulas in spreadsheet software.

Questions such as “What were the sales for the Eastern Region during the fourth quarter?” or “Are inventory levels being consistently replenished at the beginning of each week?” can easily be answered by creating metrics. Metric creation and publishing is usually the responsibility of advanced analysts.

This chapter builds on knowledge provided in the Basic Reporting Guide. You should already know how to create a simple metric and place it in a report. In this chapter you will learn the concepts necessary to create advanced metrics, including conditionality, level, metric aggregation, and metric joins.
Metric types

**Metrics** are report components that enable analytical calculations against the warehouse data. Metrics can be categorized as one of the following types, based on their formula:

- The formula of a *simple metric* is a mathematical expression based on at least one group function, such as sum or average, applied to facts, attributes, or metrics. It can also contain non-group functions or arithmetic operators, in addition to the required group function. A simple metric always has a formula and a level. The entire metric can only contain one level.

- The formula of a *compound metric* is based on arithmetic operators and non-group functions. Arithmetic operators are +, -, *, and /; non-group functions are OLAP and scalar functions such as running sum or rank. The expressions and functions can be applied to facts, attributes, or metrics.

Recall that a metric **formula** determines the data to be used and the calculations to be performed on that data.

**Simple metrics**

A simple metric does not restrict you to simple calculations; the term simple only refers to its structure. In its structure, a simple metric:

- must include at least one group function
- can include non-group functions or arithmetic operators, but not in place of the required group function
- is based on either a fact column or an attribute
- includes the specified level at which calculations are applied to the report
- can include conditions for applying calculations
- can include transformations to be done to the data prior to calculation
The following are examples of simple metrics:

\[ \text{Sum(Revenue - Cost)} \{~+\} \]
\[ \text{Sum(Abs (Revenue - Cost))} \{~+\} \]

A simple metric consists of a formula and a level. A **formula** is a mathematical expression based on at least one group function, such as sum or average, applied to facts, attributes, or metrics. A simple metric can also contain a non-group function or arithmetic operator, in addition to the required group function. However, it must be placed inside the group function, as demonstrated by the previous examples.

The level, or dimensionality, is the level of calculation for the metric, such as year or customer. Simple metrics can also contain filtering, called a condition, or offset values, called transformations. These are not required components, as are the formula and level. All of these components are discussed in detail in the section *Definition of simple metrics*.

Simple metrics have been briefly addressed in the *Metrics Essentials* chapter of the *Basic Reporting* manual. These basic metrics are generally created early in a project life cycle. They can be used on their own or as building blocks for compound metrics. An example of such a metric that calculates revenue is shown as follows:

\[ \text{Sum(Revenue)} \{~+\} \]

The \{~+\}, which is set automatically when you create a metric, means that the metrics are calculated at the lowest level on the report. For example, if the report shows revenue by year and month, the numbers are calculated to reflect monthly sales data. If an attribute is added, then that attribute is considered when the data is calculated for the report.

Simple metrics can also contain multiple facts, as in this metric definition:

\[ \text{Sum(Revenue - Cost)} \{~+\} \]

Notice that the level, represented by \{~+\}, is set on the entire metric. This concept is important to distinguish between simple and compound metrics.
Nested metrics

A nested metric provides a convenient way to use metric functionality when fact tables in the warehouse do not include attribute data at the level desired for specific analysis purposes. By using the result of a metric calculation as a temporary fact table from which to calculate another metric, you can obtain and analyze data not immediately available. For example, if you need time data aggregated at the month level, but existing fact tables provide only day-level information, you can use nested aggregation to obtain the results you are looking for.

In their structure, nested metrics:

- use the definition from another metric as part of the calculation.
- include a level definition and may also have conditions and transformations, which are independent from those of metrics being used as part of their calculation.

Although temporary tables built to calculate nested metrics are used in the same manner as other fact tables, they serve the purposes of a specific nested aggregation only; they cannot be shared.

The following is an example of a nested metric:

\[
\text{Avg}(\text{Sum}(\text{Fact})\{\sim+, \text{month+}\})\{\sim+, \text{Year+}\}
\]

The \{\sim+, \text{month+}\} dimensionality applied to the Sum metric means that the metric is calculated at the month level regardless of what appears on the report.

The \{\sim+, \text{year+}\} dimensionality applied to the Avg metric means that the metric is calculated at the year level.
Compound metrics

Compound metrics are made by combining one or more other metrics using one or more mathematical operators. The other metrics may be simple, nested, or other compound metrics. The most important distinction is that compound metrics cannot have a level placed on the entire metric, although the level can be set separately on each of the components. That is, the Revenue Metric is a simple metric defined as:

\[
\text{Sum(Revenue)} \{\sim +\}
\]

A compound metric can contain the Revenue Metric as shown below:

\[
\text{Rank([Revenue Metric])}
\]

Note that no level is set and Rank is a non-group function.

Non-group functions are OLAP and scalar functions such as rank.

A compound metric can also include expressions that act as metrics, such as:

\[
(Avg(\text{Revenue}) \{\text{Year+}\}) + (Avg(\text{Cost}) \{\text{Year+}\})
\]

Notice that while both the average functions have a level (Year), the metric as a whole does not.

Compound metrics can contain prompts and constant numerical values, but cannot include conditions, levels, or transformations except for those already part of the simple metric they contain.

Compound metrics are automatically updated when changes occur in the definitions of the metrics they include.

The parts of a compound metric are discussed in detail in the section *Definition of compound metrics.*
Derived metrics

Derived metrics are discussed in detail in the section Creating metrics in the Report Editor.

Distinguishing between simple and compound metrics

It is easy to distinguish between simple and compound metrics in the Metric Editor. Compare the following examples:

1. \([\text{Sum(Cost)} + \text{Sum(Profit)}]\)\{ReportLevel\}  
   - Formula = \(\text{Sum(Cost)} + \text{Sum(Profit)}\)  
   - Level (Dimensionality) = ReportLevel  
   - Condition = (nothing)  
   - Transformation = (nothing)

2. \(\text{Min(\text{Sum(Cost})|ReportLevel})\}\{ReportLevel\}  
   - Formula = \(\text{Min}(\text{Sum(Cost})|ReportLevel))\}  
   - Level (Dimensionality) = ReportLevel  
   - Condition = (nothing)  
   - Transformation = (nothing)

3. \(\text{Min(Revenue(Customer, ReportLevel)}\}\{ReportLevel\}  
   - Formula = \(\text{Min}(\text{Revenue(Customer, ReportLevel})\}  
   - Level (Dimensionality) = ReportLevel  
   - Condition = (nothing)  
   - Transformation = (nothing)

4. \([\text{Sum(Cost)}|ReportLevel} + \text{Sum(Profit)}|ReportLevel\}  
   - \text{Sum(Cost)}|ReportLevel\}  
   - \text{Sum(Profit)}|ReportLevel\}

Only the last example is a compound metric. The others, regardless of the complexity of their formulas, are simple metrics. When you collapse everything on a simple metric, the components (formula, level, condition, and transformation) are still visible. Since a compound metric does not contain these components at the level of the entire metric, you cannot see them. When you expand each expression of a compound metric, the components of each are exposed.
Definition of simple metrics

Metrics are constructed of components that differentiate one metric from all others and also serve as criteria for defining the calculations and data included in each metric.

Simple metrics include these components:

- The formula defines the data to be used and the calculations to be performed on the data. The outermost formula must be a group function.
- The level, or dimensionality, determines the level at which to perform the metric calculation. For example, you can choose to calculate at the month level or year level.
- Conditionality associates a filter to the metric calculation. This is an optional component.
- The transformation applies offset values, such as “four months ago,” to the selected attributes. This is also an optional component.

Recall from *Distinguishing between simple and compound metrics* that a metric defined similar to the following is a simple metric:

\[
\text{Avg} \left( \text{Sum}(\text{Revenue}) \right) \{\text{Month+}\} \{\text{Year+}\}
\]

This metric calculates the yearly average of monthly sales, and is actually two metrics. In the metric definition pane of the Metric Editor, the inner metric is contained within the outer metric. To view the definition of the inner metric, you must expand the formula of the outer metric. It is a simple metric because it can contain a level, condition, and transformation at its highest level, as illustrated by this screen shot from the Metric Editor:
The inner metric is \( \text{Sum}([\text{Revenue}]) \). Recall that this was previously defined as a simple metric. The code \{Month\} within the same set of parentheses indicates that this data is tallied at the month level, regardless of what appears on the report. Once this information is calculated, the second metric is performed, that is, the result from the first metric is averaged at the year level. The following diagram represents this process.

This type of metric provides a convenient method for multistep calculations when fact tables in the warehouse do not provide data at the appropriate higher level for subsequent calculations. You can therefore use the result of a metric calculation as an intermediate result set for calculating another metric. For example, your existing fact tables provide revenue figures for each day, but you need monthly sales information. Using this kind of metric allows you to obtain the monthly figures.

**Formula**

This is the essential part of the metric definition. The **formula** of a simple metric is a mathematical expression consisting of group functions, such as sum, average, minimum, maximum, and so on. It also includes the data to be used in the calculations. This can include facts, attributes, constants, and other metrics. The formula can also contain a non-group function or arithmetic operator, in addition to the required group function.

In SQL, the formula becomes part of the SELECT clause of the SQL command.
Defining custom plug-in functions

The MicroStrategy Function Plug-In Wizard can be used for defining custom functions relevant to your business case scenarios. The Intelligence Server makes no distinction between these custom functions and the ones provided by default. These custom plug-in functions are indistinguishable from all other functions or operators, such as $\text{Sum}$, $\text{Average}$, $\text{Min}$, $\text{Max}$, $\text{Count}$, $-$, $+$, $/$, or $\ast$.

Defining custom plug-in functions involves the following steps:

- In the **design** stage, you determine how to implement the analytical procedures into a computer system.

- **Creation** builds the Microsoft Visual C++ project, which is used to produce a library containing your algorithms.

- **Implementation** involves creating the code for the algorithms and compiling this code into a library that will be used by MicroStrategy.

- **Importing** adds the library to a MicroStrategy project so that its algorithms are available for use in the project.

- **Deployment** distributes your library to the MicroStrategy Intelligence Servers, which will execute it.

- The final step is **execution**, which is creating new metrics that use the algorithms and using those metric in a MicroStrategy report.

The Function Plug-In Wizard guides you through the creation and implementation steps. It helps you create a Microsoft Visual C++ project with placeholders where you can add custom analytic code. After adding your function-specific C++ code and building your project, you can launch MicroStrategy Desktop to import your new function plug-in to be used for all the reports. Deployment occurs on each Intelligence Server system that will use it. The execution step is also performed in MicroStrategy Desktop, when you create metrics and reports using the new function. For detailed information on each step, see the Function Plug-In Wizard online help. The MicroStrategy online help also provides instructions on importing the functions.
The Function Plug-In Wizard must be installed and activated before you can create and implement custom plug-in functions. The self-extracting installation file, named FPWizard.exe, is located in the \MicroStrategy\Desktop directory. For installation and activation procedures, see the MicroStrategy Functions Reference.

Base formulas

You can recycle a formula to use it in multiple metric definitions. This is called a base formula, which can contain arithmetic operators, attributes, facts, group functions, metrics, and non-group functions. Using a base formula allows you to

- update multiple metrics by modifying the base formula only once, as the change is automatically propagated to all the metrics that use the base formula
- find or categorize all the metrics that use a common base formula
- use a simple expression as a base formula to facilitate the creation of more complex metrics
- use it as a formula in simple or compound metrics

Level

The level of a metric, also referred to as dimensionality, allows you to determine the attribute level at which the metric is calculated. In addition, you can specify the grouping and filtering involved in a metric.

The concepts underlying the term level in the context of MicroStrategy metrics are interchangeable with those of dimensionality. The term level is used throughout this manual.
All metrics, by default, calculate at the report level. This means that the attributes on the report template dictate how the metric is aggregated. For example, if the report shows revenue by year and month, the numbers are calculated to reflect monthly sales data. However, you can specify any set of attributes as the calculation level of a metric. The engine determines the set that is at the lowest level; for example, Region, Month, and Year resolves to Region and Month.

By default, the report level is part of the metric level. This allows for more flexibility in the use of the metric.

The elements needed to specify a level for a particular metric are:

- **target**
- **grouping**
- **filtering**

A target, grouping, and filtering combination composes one level unit. Grouping and filtering are independent of each other. That is, the target and grouping work together to determine the level, and the target and filtering also work together to establish the level.

Clicking Reset on the Level pane of the Metric Editor changes the level unit back to the default of report level for the target and standard for filtering and grouping.

**Target**

The *target* is the attribute level at which the metric calculation groups. It determines the table to use to calculate the metric. Any set of attributes or a hierarchy can be the target. A special case is the default target, which is report level.
**Grouping**

*Grouping* determines how the metric aggregates. The result of this setting is reflected in the **GROUP BY** clause of the SQL command. The grouping options for levels include:

- Standard groups by the attribute level of the target. That is, the metric calculates at the level of the target, if possible.

- None excludes the attribute in the target from the **GROUP BY** clause. It also excludes any of the target attribute’s children.

None is not an option if the target is set to the report level.

The remaining options are only used for nonaggregatable metrics. A **nonaggregatable metric** is one that should not be aggregated across an attribute. An example is an inventory metric. While the data warehouse may record the inventory every month, these monthly numbers are not added together to calculate the yearly inventory. Instead, you may want to use the End-On-Hand and Beginning-On-Hand inventory numbers to see how the total inventory changed over the year. These grouping options, described below, are used in such cases:

- Beginning lookup uses the first value of the lookup table.

- Ending lookup uses the last value of the lookup table.

- Beginning fact accesses the first value of the fact table.

- Ending fact accesses the last value contained in the fact table.

Setting a metric level to one of the options listed above defines the metric as nonaggregatable. Whether you select a fact or lookup table largely depends on how the necessary information is stored. For example, to find the Beginning-on-Hand inventory for a particular item, you need to know how the inventory information is stored. If the inventory count is not taken on the first day of the week, as the lookup table requires, the inventory count should be taken from the fact table for the first recorded entry.
There is another important difference between accessing a fact table and a lookup table. If a value, such as April sales, is missing from a fact table, the row still exists in the table and is reported as null or zero. If that same value is missing in a lookup table, the April row does not exist. The previous or next value (March or May) is reported, depending on whether the level is set to beginning or ending value.

**Level grouping examples**

A revenue metric is defined as:

\[ \text{Sum(Revenue)} \{\text{Quarter}\} \]

The level target is set to quarter, with standard grouping. When this metric is placed on a report with quarter, the results are shown below.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Metrics</th>
<th>GroupByQuarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
<tr>
<td>Q2 02</td>
<td>$2,585,002</td>
<td></td>
</tr>
<tr>
<td>Q3 02</td>
<td>$1,545,399</td>
<td></td>
</tr>
<tr>
<td>Q4 02</td>
<td>$2,861,785</td>
<td></td>
</tr>
<tr>
<td>Q1 03</td>
<td>$1,222,374</td>
<td></td>
</tr>
<tr>
<td>Q2 03</td>
<td>$2,302,652</td>
<td></td>
</tr>
<tr>
<td>Q3 03</td>
<td>$1,560,244</td>
<td></td>
</tr>
<tr>
<td>Q4 03</td>
<td>$2,894,534</td>
<td></td>
</tr>
</tbody>
</table>

Notice that the sales are calculated for each quarter, because the metric is grouping at the quarter level, as shown in the SQL:

```sql
select a11.[QUARTER_ID] AS QUARTER_ID,
    max(a12.[QUARTER_DESC]) AS QUARTER_DESC,
    sum(a11.[TOT_DOLLAR_SALES]) as REVENUE
from [QTRCATEGORY_SLS] a11,
    [LUQUARTER] a12
where a11.[QUARTER_ID] = a12.[QUARTER_ID]
```
Using the same metric on a report with month, however, yields the following results.

<table>
<thead>
<tr>
<th>Month</th>
<th>Metrics</th>
<th>GroupByQuarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
<tr>
<td>Feb 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
<tr>
<td>Mar 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
<tr>
<td>Apr 02</td>
<td>$2,585,002</td>
<td></td>
</tr>
<tr>
<td>May 02</td>
<td>$2,585,002</td>
<td></td>
</tr>
<tr>
<td>Jun 02</td>
<td>$2,585,002</td>
<td></td>
</tr>
</tbody>
</table>

This is only a subset of the report.

Although each month is listed, the value for each month in a quarter is the same. The metric is calculating quarterly revenue, based on the grouping level set on the metric. The SQL for this report is, in essence, the same as the previous example:

```sql
insert into TEMP_TABLE
select a11.[QUARTER_ID] AS QUARTER_ID,
     sum(a11.[TOT_DOLLAR_SALES]) as REVENUE
from [QTR_CATEGORY_SLS] a11
group by a11.[QUARTER_ID]

select a11.[MONTH_ID] AS MONTH_ID,
     a11.[MONTH_DESC] AS MONTH_DESC,
     pa1.[REVENUE] as REVENUE
from [TEMP_TABLE] pa1,
     [LU_MONTH] a11
where pa1.[QUARTER_ID] = a11.[QUARTER_ID]
```
Change the grouping to none on that same revenue metric and place it on a report with year. Because year is a parent of quarter, the metric can roll up to the year level. The report and its SQL are illustrated below.

```
select a12.[YEAR_ID] AS YEAR_ID,
       sum(a11.[TOT_DOLLAR_SALES]) as REVENUE
from [QTR_CATEGORY_SLS] a11,
     [LU_QUARTER] a12
where a11.[QUARTER_ID] = a12.[QUARTER_ID]
group by a12.[YEAR_ID]
```

A total year sales fact table exists in the project, which would be more efficient. Instead of adding all the quarters together, the yearly total could have been pulled directly from that table. However, having quarter in the level of the metric forces the report to use the quarter sales table.

If the same revenue metric, with the grouping set to none, is used on a report with month, the results are shown below.

```
Month | Metrics | Don't Group By Quarter
----- | ------- | ---------------------
Jan 02 | $16,936,434
Feb 02 | $16,936,434
Mar 02 | $16,936,434
Apr 02 | $16,936,434
May 02 | $16,936,434
Jun 02 | $16,936,434
Jul 02 | $16,936,434
Aug 02 | $16,936,434
Sep 02 | $16,936,434
```
The metric calculates the same number for each month—the total for all the months included on the report. Because month is a child of quarter, month is excluded from the group by clause:

```sql
insert into TEMP_TABLE
select sum(a11.[TOT_DOLLAR_SALES]) as REVENUE
from [QTR_CATEGORY_SLS] a11

select a11.[MONTH_ID] AS MONTH_ID,
    a11.[MONTH_DESC] AS MONTH_DESC,
    pa1.[REVENUE] as REVENUE
from [TEMP_TABLE] pa1,
    [LU_MONTH] a11
```

Inventory is one example of a nonaggregatable metric. The following metric definition reports the inventory on hand at the end of the month. The level is set at the report level and at month, with a grouping of ending fact, so that the last entry in the fact table is used.

```
Sum([End on hand]) {~, Month}
```

A report contains this metric and the month attribute. The last entry for each month in the fact table is placed on the report. No calculation is performed.
When the same metric is used on a report with quarter, the value for each quarter is the value for the last month in the quarter. The monthly values for each quarter are not added together. For example, the on-hand inventory for March 2002 is 33,740. Since that is the last month in Q1, that value is reported on the quarterly report.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>33,740</td>
</tr>
<tr>
<td>Q2 02</td>
<td>30,320</td>
</tr>
<tr>
<td>Q3 02</td>
<td>38,499</td>
</tr>
<tr>
<td>Q4 02</td>
<td>28,659</td>
</tr>
<tr>
<td>Q1 03</td>
<td>38,626</td>
</tr>
<tr>
<td>Q2 03</td>
<td>32,295</td>
</tr>
<tr>
<td>Q3 03</td>
<td>43,697</td>
</tr>
<tr>
<td>Q4 03</td>
<td>28,576</td>
</tr>
</tbody>
</table>

This is only a sample of the report, not the entire report.
Filtering

The filtering setting governs the relationship between the report filter and the calculation of the metric. The filtering options are:

- Standard filtering allows the report filter to interact as usual in the metric calculation. The metric calculates only for the elements found in the filter definition. The filter criteria for the report is found in the WHERE clause of the SQL statement which calculates the metric in question.

- Absolute filtering changes the filter on descendents of the target. It raises it to the level of the target, if possible.
  - If the attribute in the metric filter is a parent of the attribute in the report filter, calculations are performed only on elements to which the report filter applies.
  - If the attribute in the metric filter is of the same level or a child of the level in the report filter, calculations occur as specified by the report filter.

Absolute filtering influences what is displayed on the report, not its calculations. It includes the report criteria in a subquery rather than in the WHERE clause itself.

- Ignore filtering omits filtering criteria based on the attribute in the target and its related attributes (parents and children). The report filter does not appear anywhere in the SQL for a metric with this setting.

- None can be summarized as unspecified—the filtering behavior for the target is not determined by this component. Instead, the target and group components of this level unit define the filter.
  - If the report includes an attribute in the same hierarchy as that indicated by the metric filter, aggregation takes place at the level of that attribute.
  - If the report does not include other attributes in the same hierarchy as that indicated by the metric filter, aggregation defaults to the “Absolute” option.
Level filtering examples

Consider the following report as a baseline to show the revenue for each month and quarter.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Month</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>Jan 02</td>
<td></td>
<td>$514,996</td>
</tr>
<tr>
<td></td>
<td>Feb 02</td>
<td></td>
<td>$579,508</td>
</tr>
<tr>
<td></td>
<td>Mar 02</td>
<td></td>
<td>$789,940</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$1,884,444</td>
</tr>
<tr>
<td>Q2 02</td>
<td>Apr 02</td>
<td></td>
<td>$913,442</td>
</tr>
<tr>
<td></td>
<td>May 02</td>
<td></td>
<td>$901,618</td>
</tr>
<tr>
<td></td>
<td>Jun 02</td>
<td></td>
<td>$769,942</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$2,585,002</td>
</tr>
<tr>
<td>Q4 03</td>
<td>Oct 03</td>
<td></td>
<td>$640,002</td>
</tr>
<tr>
<td></td>
<td>Nov 03</td>
<td></td>
<td>$957,865</td>
</tr>
<tr>
<td></td>
<td>Dec 03</td>
<td></td>
<td>$1,296,667</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$2,894,534</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>$16,936,434</td>
</tr>
</tbody>
</table>

All of the metrics in these examples have grouping set to none. None of the reports are presented in full; they are only subsets of the complete report.

A revenue metric is defined with quarter as the target and standard filtering. A report is created with month, quarter, this new revenue metric, and a filter for January 2002. When the report is executed, the revenue is the same for every row, as shown below. All months are included on the report, even though the report filter is January 2002. This is an effect of setting the grouping to none. Since quarter in the target is a parent of month in the filter, all months are included on the report. The metric value is the grand total of the filter, in this case, January 2002 only.
The same report is created with a metric set to absolute filtering. When the report is executed, the revenue is the same for every row, as shown below. Because of the absolute setting, the report filter rolls up to the level of the metric—that is, month is elevated to quarter. Because the report is filtered for January 2002, the value is revenue for the first quarter of 2002.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Month</th>
<th>Metrics</th>
<th>Filter By Quarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>Jan 02</td>
<td>$849,529</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feb 02</td>
<td>$849,529</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mar 02</td>
<td>$849,529</td>
<td></td>
</tr>
<tr>
<td>Q2 02</td>
<td>Apr 02</td>
<td>$849,529</td>
<td></td>
</tr>
<tr>
<td></td>
<td>May 02</td>
<td>$849,529</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jun 02</td>
<td>$849,529</td>
<td></td>
</tr>
</tbody>
</table>

The same report is run, but this time with a metric that has level filtering set to ignore. Again the metric value is the same for the whole report, but now it is the grand total of all sales in the project. Since month is related to quarter, the filter is also ignored.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Month</th>
<th>Metrics</th>
<th>Absolute By Quarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>Jan 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feb 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mar 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
<tr>
<td>Q2 02</td>
<td>Apr 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
<tr>
<td></td>
<td>May 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jun 02</td>
<td>$1,884,444</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Month</th>
<th>Metrics</th>
<th>Ignore By Quarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 02</td>
<td>Jan 02</td>
<td>$16,936,434</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feb 02</td>
<td>$16,936,434</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mar 02</td>
<td>$16,936,434</td>
<td></td>
</tr>
<tr>
<td>Q2 02</td>
<td>Apr 02</td>
<td>$16,936,434</td>
<td></td>
</tr>
<tr>
<td></td>
<td>May 02</td>
<td>$16,936,434</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jun 02</td>
<td>$16,936,434</td>
<td></td>
</tr>
</tbody>
</table>
Advanced options for metric levels

The advanced options for metric levels include the following settings:

- **Allow other users to add extra units to this definition**, which is used to emulate MicroStrategy 6.x behavior, affects only those projects that have been upgraded from 6.x. The option indicates whether the metric accepts dimensionality units, for metrics used at the template level and metrics used in the filter for a metric qualification. This continuation dimensionality is merged with the original units to complete the metric level.

- **Add attributes in the filter to the level (dimensionality)** determines whether the metric filter is applied to the metric calculation. By default, the setting is selected. If it is cleared, filter attributes that are not on the template or the level are not included in the metric calculation.

Add filter attributes to the level example

The best way to explain the Add attributes in the filter to the level setting is with an example. The following reports all contain first quarter revenue for books sold in California stores, but depending on the Add attributes setting, that revenue amount changes.

1. Create a filter with the following conditions and name it **CA Books Q1 2002**:
   - Call Center = San Diego and San Francisco
   - Category = Books

2. Create a revenue metric and use the CA Books Q1 2002 filter as the condition. By default, the Add attributes setting is selected. Name it **Revenue (Attributes On)**.
3 Copy the Revenue (Attributes On) metric, renaming it Revenue (Attributes Off). Edit the metric to clear the Add attributes setting, by following the steps outlined below:

- Select Level (Dimensionality) in the breakdown window (under the heading Metric is defined as). The Definition window changes to display level options.
- Click Advanced in the Definition window. The Level (Dimensionality) advanced options dialog box opens.
- Clear the Add attributes in the filter to the level (dimensionality) check box.
- Click OK to return to the Metric Editor.
- Click Save and Close to return to the Desktop.

4 Create a report with the Region and Call Center attributes on the rows and the Revenue (Attributes On) metric on the columns. Execute the report. The results are displayed below:

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Revenue (Attributes On)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northwest</td>
<td>San Francisco</td>
<td>515</td>
</tr>
<tr>
<td>Southwest</td>
<td>San Diego</td>
<td>849</td>
</tr>
</tbody>
</table>

5 Change to SQL view and notice the Where clause, as shown below:

```sql
where a11.[SUBCAT_ID] = a12.[SUBCAT_ID] and a11.[CALL_CTR_ID] = a13.[CALL_CTR_ID] and a13.[REGION_ID] = a14.[REGION_ID] and (a11.[CALL_CTR_ID] in (2, 4) and a12.[CATEGORY_ID] in (1) and a11.[MONTH_ID] in (200201, 200202, 200203))
```

The complete metric filter (Call Center, Category, and Month, as shown in bold font) is included in the metric calculation.

6 Save the report as CA Revenue (Attributes On).
7  Return to Design view. Delete the Revenue (Attributes On) metric and replace it with the Revenue (Attributes Off) metric. Execute the report. The results are displayed below:

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Revenue (Attributes Off)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northwest</td>
<td>San Francisco</td>
<td>1,050,983</td>
</tr>
<tr>
<td>Southwest</td>
<td>San Diego</td>
<td>2,397,919</td>
</tr>
</tbody>
</table>

8  Why has the revenue increased substantially? Change to SQL view to check the Where clause:

\[
\text{where a11.[CALL_CTR_ID] = a12.[CALL_CTR_ID]}
\]
\[
\text{and a12.[REGION_ID] = a13.[REGION_ID]}
\]
\[
\text{and a11.[CALL_CTR_ID] in (2, 4)}
\]

With the Add attributes setting turned off, only those attributes in the metric filter which are on the template or in the metric level are included in the metric calculation. In this report, only Call Center, as shown in bold above, meets those requirements, since it is on the template. Because the metric conditions of Category = Book and Month = January - March '00 are excluded, the revenue is calculated for all categories and all time, increasing the revenue amount dramatically.

In the previous examples, the metric level has not changed from the default of report level, so it does not really affect the Add attributes setting. The next example adds a metric level.

9  Save the report as CA Revenue (Attributes Off).

10 Copy the Revenue (Attributes Off) metric, renaming it Yearly Revenue (Attributes Off). Edit the metric to add Year to the metric level.

11 Copy the CA Revenue (Attributes Off) report, renaming it Yearly CA Revenue (Attributes Off).
12 Edit the new report. Delete the Revenue (Attributes Off) metric and replace it with the Yearly Revenue (Attributes Off) metric. Execute the report. The results are displayed below:

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northwest</td>
<td>San Francisco</td>
<td>153,085</td>
</tr>
<tr>
<td>Southwest</td>
<td>San Diego</td>
<td>288,073</td>
</tr>
</tbody>
</table>

13 The revenue amount has changed again. Check the Where clause in the SQL view to discover why:

```sql
where a11.[DAY_DATE] = a12.[DAY_DATE] and
    a11.[CALL_CTR_ID] = a13.[CALL_CTR_ID] and
    a13.[REGION_ID] = a14.[REGION_ID]
    and(a11.[CALL_CTR_ID] in (2, 4)
    and a12.[MONTH_ID] in (200201, 200202, 200203))
```

14 Now the metric calculation includes Call Center because it is defined on the template and Month because it is in the same hierarchy as Year, which is the metric level. Category is not included, since it is neither on the template or in the metric level. The metric calculates revenue in all categories for the California stores for the first quarter of 2002.
Example 1: Using level metrics

Report requirements

Your company has recently kicked off a new ad campaign targeted at certain areas that present high growth opportunities. In your regions, this consists of the Boston, New York, and the Washington DC call centers. You need to perform an analysis from different perspectives and are looking for answers to the following:

1. How do the sales of each call center compare to the total sales of the targeted call centers in a given region?
2. How do the sales of each call center compare to the total sales of all the call centers in a given region?
3. How do the sales of each call center compare to the total sales of all the call centers in a given region for a given year?

Solution 1

Grouping set to Standard and Filtering set to Standard

In this case, the Regional Sales is equal to the sum of the revenues of the call centers in a given region. This sum takes into account only those call centers that are included in the report filter. For example, the Mid-Atlantic Regional Sales only includes the Washington DC call center sales as this is the only call center from that region that has been included in the report filter. The metric groups at the target level of Region because grouping is standard.

With standard filtering, all of the report filter elements are included in the calculation of the metric. This occurs by placing the report filter in the WHERE clause of the SQL pass for this metric that is shown in the following example:
Example 1: Using level metrics

© 2005 MicroStrategy, Inc.

```sql
sum(a11.[ORDER_AMT]) as REGIONALSALES
from [ORDER_FACT] a11,[LU_EMPLOYEE]a12,
[LU_CALL_CTR] a13
where a11.[EMP_ID] = a12.[EMP_ID]
and a12.[CALL_CTR_ID] = a13.[CALL_CTR_ID]
and a12.[CALL_CTR_ID] in (5, 11, 12)
group by a13.[REGION_ID]
```

The report is displayed as follows:

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Regional Sales (Target=Region, Filtering=Standard, Grouping=Standard)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Boston</td>
<td>$1,325,448</td>
<td>$2,334,864.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>$1,009,416</td>
<td>$2,334,864.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$2,334,864</td>
<td>$2,334,864.00</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Washington, DC</td>
<td>$1,413,865</td>
<td>$1,413,865.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$1,413,865</td>
<td>$1,413,865.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$3,748,729</td>
<td>$3,748,729.00</td>
<td></td>
</tr>
</tbody>
</table>

The Revenue subtotals match up with the values of the total Regional Sales.

**Solution 2**

**Grouping set to Standard and Filtering set to Absolute**

In this case, the Regional Sales is equal to the sum of revenues of all call centers included in a given region. Grouping continues to occur at the target attribute level of Region.
With absolute filtering, the report filter is present in the subquery of the WHERE clause in the SQL pass for this metric as shown in the following example:

```sql
select a13.[REGION_ID]) as REGION_ID,
    sum(a11.[ORDER_AMT]) as REGIONALSALES
from [ORDER_FACT] a11,[LU_EMPLOYEE]a12,
    [LU_CALL_CTR] a13
where a11.[EMP_ID] = a12.[EMP_ID]
and a12.[CALL_CTR_ID] = a13.[CALL_CTR_ID]
and ((a13.[REGION_ID])
in (select s21.[REGION_ID]
    from [LU_CALL_CTR] s21
    where s21.[CALL_CTR_ID] in (5,11,12)))
group by a13.[REGION_ID]
```

The report is shown in the following figure:

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Regional Sales (Target=Region, Filtering=Absolute, Grouping=Standard)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Boston</td>
<td></td>
<td>$ 1,325,448</td>
<td>$2,334,864.00</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td></td>
<td>$ 1,009,416</td>
<td>$2,334,864.00</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$ 2,334,864</td>
<td>$2,334,864.00</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Washington, DC</td>
<td></td>
<td>$ 1,413,865</td>
<td>$3,413,340.00</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$ 1,413,865</td>
<td>$3,413,340.00</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$ 3,748,729</td>
<td>$5,748,204.00</td>
</tr>
</tbody>
</table>
Note the following:

- With absolute filtering, the report filter is placed in the subquery of the WHERE clause only if it is of a lower level than the target. If the report filter is of a higher level than the target, there is no need for a subquery and so the engine does not use one.

- The VLDB properties of the report may be changed to use two passes of SQL rather than a subquery. VLDB properties are discussed later in this chapter.

**Solution 3**

**Grouping set to Standard and Filtering set to Ignore**

In this case, the engine ignores the report filter and the report displays the Regional Sales as the sum of revenues of all the call centers in that region.

With no filtering, the report filter elements that are directly related to the target attributes are not placed in the WHERE clause of the SQL pass for the metric as shown in the following example:

```sql
select a13.[REGION_ID]) as REGION_ID,
       sum(a11.[ORDER_AMT])as REGIONALSALES
from [ORDER_FACT] a11,[LU_EMPLOYEE]a12,
     [LU_CALL_CTR]a13
where a11.[EMP_ID] = a12.[EMP_ID]
and a12.[CALL_CTR_ID] = a13.[CALL_CTR_ID]
group by a13.[CALL_CTR_ID]
```

If the report filter contains attribute elements such as Year, these attributes are not ignored because they are not directly related to the target attribute Region.
In the following example, since call centers are directly related to the target attribute Region, the entire report filter is ignored. The report displays Revenue as the sum of revenues for the years 2002 and 2003 and Regional Sales as sum of revenues for the years 2002 and 2003.

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Regional Sales (Target=Region, Filtering=Ignore, Grouping=Standard)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Boston</td>
<td>$1,325,448</td>
<td>$2,334,864.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>$1,009,416</td>
<td>$2,334,864.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$2,334,864</td>
<td>$2,334,864.00</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Washington, DC</td>
<td>$1,413,865</td>
<td>$3,413,340.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$1,413,865</td>
<td>$3,413,340.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$3,748,729</td>
<td>$5,748,204.00</td>
<td></td>
</tr>
</tbody>
</table>
In the example that follows, Year 2003 is included in the report filter. As a result, the engine ignores the report filter but calculates Revenue and Regional Sales for the year 2003 only.

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Regional Sales (Target=Region, Filtering=Ignore, Grouping=Standard)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Boston</td>
<td>$628,689</td>
<td>$1,103,875.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>$475,186</td>
<td>$1,103,875.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$1,103,875</td>
<td>$1,103,875.00</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Washington, DC</td>
<td>$666,625</td>
<td>$1,628,718.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$666,625</td>
<td>$1,628,718.00</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>$1,770,500</td>
<td>$2,732,593.00</td>
<td></td>
</tr>
</tbody>
</table>

Security filters are included in the WHERE clause of the level metric’s SQL statement even with absolute or ignore filtering. The engine includes the security filter to ensure that there is no breach in security for any level metric. With filtering ignored, the security filter is unioned with the report filter and is applied to the metric also. With absolute filtering, the security filter is applied in the subquery with the report filter.
Example 2: Using level metrics

Report requirements

Your company has recently kicked off a new ad campaign targeted at certain areas that present high growth opportunities. In your regions, this consists of the Boston, New York, and the Washington DC call centers. You need to perform an analysis from different perspectives and are looking for answers to the following:

1. How did the sales of these three call centers compare to the total of all three?
2. How did the sales of these three call centers compare to the total sales of all call centers within the targeted regions?
3. How did the sales of each of the three call centers compare to the sales of the entire company?
4. What were the sales in each region, based on the items sold in each call center in that region?

The answers to these questions give you an insight into how the new campaign is being received in the targeted areas of your region.

Solution 1

Grouping set to None and Filtering set to Standard

In this business scenario, the Regional Sales metric calculates the total sales for all call centers present in the report filter. By changing grouping to none, the metric does not group by anything directly related to the target attribute specified within the metric.
Therefore, in this example, there is no GROUP BY statement in the SQL as the attributes Call Center and Region are directly related to the metric target Region. With standard filtering, the report filter elements are included in the WHERE clause of the SQL as shown in the following example:

```sql
select sum(a11.[ORDER_AMT])as REGIONALSALES
from [ORDER_FACT] a11,[LU_EMPLOYEE]a12
where a11.[EMP_ID] = a12.[EMP_ID]
and a12.[CALL_CTR_ID]in(5,11,12)
```

The report is displayed in the following figure:

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Regional Sales (Target=Region, Filtering=Standard, Grouping=none)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Boston</td>
<td>$1,325,448</td>
<td>$3,748,729.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>$1,000,416</td>
<td>$3,748,729.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$2,334,864</td>
<td>$3,748,729.00</td>
<td></td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Washington, DC</td>
<td>$1,413,865</td>
<td>$3,748,729.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$1,413,865</td>
<td>$3,748,729.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$3,748,729</td>
<td>$3,748,729.00</td>
<td></td>
</tr>
</tbody>
</table>

**Solution 2**

**Grouping set to None and Filtering set to Absolute**

In this scenario, the Regional Sales metric calculation includes the total for all the call centers present within all the regions listed in the report, and not just the call centers included in the report filter.
With no grouping, the metric does not group by anything directly related to the target attribute specified within the metric. Since the attributes Region and Call Center in this example are related to the target, there is no GROUP BY clause in the SQL as shown in the following example:

```sql
select sum(a11.[ORDER_AMT])as REGIONALDOLL
from [ORDER_FACT] a11,[LU_EMPLOYEE]a12,
    [LU_CALL_CTR]a13
where a11.[EMP_ID] = a12.[EMP_ID]
and a12.[CALL_CTR_ID] = a13.[CALL_CTR_ID]
and ((a13.[REGION_ID])
in (select s21.[REGION_ID]
from [LU_CALL_CTR] s21
where s21.[CALL_CTR_ID] in (5,11,12)))
```

Also, with absolute filtering, the report filter is placed in the subquery only if it is of a lower level than the target. The report is shown in the following figure:
Solution 3

Grouping set to None and Filtering set to Ignore

The Regional Sales metric calculates the total company sales for all call centers, ignoring the three call centers that are filtered out in the report filter.

With no grouping, the metric does not group by anything directly related to the target attribute specified within the metric. Since the attributes Region and Call Center are related to the target, there is no GROUP BY clause in the SQL as shown in the following example:

```sql
select sum(all.[TOT_DOLLAR_SALES]) as REGIONALSALES
from [YRCATEGORY_SLS] all
```

The report is shown in the following figure:

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Metrics</th>
<th>Revenue</th>
<th>Regional Sales (Target=Region, Filtering=Ignore, Grouping=None)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Boston</td>
<td></td>
<td>$1,325,448</td>
<td>$16,936,434.00</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td></td>
<td>$1,009,416</td>
<td>$16,936,434.00</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$2,334,864</td>
<td>$16,936,434.00</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Washington, DC</td>
<td></td>
<td>$1,413,865</td>
<td>$16,936,434.00</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$1,413,865</td>
<td>$16,936,434.00</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>$3,748,729</td>
<td>$16,936,434.00</td>
</tr>
</tbody>
</table>
Solution 4

Grouping set to None and Filtering set to None

The Regional Sales metric calculates the total sales based on the number of items sold in each call center.

With no grouping, there is no GROUP BY clause for this metric calculation. With no filtering, you can define a fact of your choice in the calculation of a metric. This is accomplished by adding as many additional target attributes as necessary to the metric to force it to use the fact table that you want. Any target attribute that has no filtering borrows its filtering criteria from the other target attributes specified in the dimensionality of the metric. This allows you to choose the fact table but not alter the original intent of the report. The SQL statements for this example are as follows:

Regional Sales (Target=Region, Filtering=Standard, Grouping=Standard)

```
select a12.[REGION_ID] as REGION_ID,
    sum((a11.[QTY_SOLD] * a11.[UNIT_PRICE]-a11.[DISCOUNT])) as REGIONALDOLL
from [ORDER_FACT] a11, [LU_CALL_CTR]a12, [LU_EMPLOYEE]a13
where a11.[EMP_ID] = a12.[EMP_ID]
    and a12.[CALL_CTR_ID] = a13.[CALL_CTR_ID]
    and a11.[CALL_CTR_ID] in (5,11,12)
    group by a12.[REGION_ID]
```

Regional Sales1 (Target1=Region, Filtering=Standard, Grouping=Standard, Target2=Item, Filtering=None, Grouping=Standard)

```
select a12.[REGION_ID] as REGION_ID,
    sum((a11.[QTY_SOLD] * (a11.[UNIT_PRICE]-a11.[DISCOUNT]))) as REGIONALDOLL
```

```sql
from [ORDER_DETAIL] a11, [LU_CALL_CTR] a12
where a11.[EMP_ID] = a12.[EMP_ID]
and a11.[CALL_CTR_ID] = a12.[CALL_CTR_ID]
and a11.[CALL_CTR_ID] in (5, 11, 12)
group by a12.[REGION_ID]
```

In this business scenario, if you want to use the Order_Detail fact table instead of the Order_Fact table, you include the Item attribute as the target. Since the Item attribute is found in the Order_Detail table and not in the Order_Fact table, it forces the engine to use the Order_Detail fact table. The report is displayed in the following figure:

<table>
<thead>
<tr>
<th>Region</th>
<th>Call Center</th>
<th>Metrics</th>
<th>Regional Sales (Target=Region, Filtering=Standard, Grouping=Standard)</th>
<th>Regional Sales1 (Target1=Region, F=St, G=St, Target2=Item, F=None, G=None)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td>Boston</td>
<td>Revenue</td>
<td>$1,325,448</td>
<td>$2,334,864.00</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td></td>
<td>$1,009,416</td>
<td>$2,334,864.00</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td>Washington, DC</td>
<td></td>
<td>$1,413,865</td>
<td>$1,413,865.00</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>$3,748,729</td>
<td>$3,748,729.00</td>
</tr>
</tbody>
</table>

In this example, Regional Sales is calculated using both the Order_Fact table and the Order_Detail fact table just to show that the data in the Order Detail fact table adds up correctly to match the data in the Order fact table.
Example 3: Removing report level

Report requirement

You want to compare the sales performance of the targeted areas to the total sales in the company everywhere and for all time.

Solution

To display this report, remove the report level target that is present by default, and add any attribute as the target with no grouping, rather than including multiple attribute levels in one metric.

By removing the report level from the target and with no grouping for any other available attribute, there is no GROUP BY clause in the SQL. Any attribute can be used for this purpose. There is no need to add more than one attribute, unless a special filtering behavior is required for the metric. If a special filtering behavior is required, then other attributes are required but they should not be grouped.

This is a quick and easy way to do something that might otherwise involve multiple steps. It is especially helpful if you have many dimensions represented on a report that need to be included in the metric calculation in order to obtain the desired outcome.
Condition

 Metric conditionality applies a filter to the metric calculation. You can think of conditionality as attaching a filter to each metric. For example, a report shows revenue by region and uses the revenue metric. Revenue for all years is included on the report. If you associate a 2003 filter to the revenue metric and re-execute the report, the report displays data for 2003 only.

 Only metrics with an aggregate operator in the formula can use conditionality. Only one filter can be associated with a simple metric, although that filter can contain as many filtering conditions as needed.

 To create a report that compares the monthly sales to January sales, define the following metrics:

 1. **Revenue (report level)**
     
     \[
     \text{Sum(Revenue) \{\sim\}}
     \]

 2. **January Revenue (level of Month of Year, standard filtering, no grouping; condition of January)**
     
     \[
     \text{Avg(Revenue) \{\sim, [Month of Year]\} <January>}
     \]

     Consider Revenue as a metric for calculating January Revenue

 3. **Monthly Revenue (level of Month, standard filtering, standard grouping)**
     
     \[
     \text{Sum(Revenue) \{Month\}}
     \]

 4. **Variance from January**
     
     \[
     ([\text{Monthly Revenue}] - [\text{January Revenue}])
     \]
Place these metrics on a report with Month to achieve the following report:

<table>
<thead>
<tr>
<th>Month</th>
<th>Metrics</th>
<th>Revenue</th>
<th>January Revenue</th>
<th>Variance From January</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 02</td>
<td>$514,996.00</td>
<td>$424,764.50</td>
<td>$90,231.50</td>
<td></td>
</tr>
<tr>
<td>Feb 02</td>
<td>$579,508.00</td>
<td>$424,764.50</td>
<td>$154,743.50</td>
<td></td>
</tr>
<tr>
<td>Mar 02</td>
<td>$789,940.00</td>
<td>$424,764.50</td>
<td>$365,175.50</td>
<td></td>
</tr>
<tr>
<td>Apr 02</td>
<td>$913,442.00</td>
<td>$424,764.50</td>
<td>$488,677.50</td>
<td></td>
</tr>
<tr>
<td>May 02</td>
<td>$901,618.00</td>
<td>$424,764.50</td>
<td>$476,053.50</td>
<td></td>
</tr>
<tr>
<td>Jun 02</td>
<td>$769,942.00</td>
<td>$424,764.50</td>
<td>$345,177.50</td>
<td></td>
</tr>
<tr>
<td>Jul 02</td>
<td>$346,220.00</td>
<td>$424,764.50</td>
<td>-$78,544.50</td>
<td></td>
</tr>
<tr>
<td>Aug 02</td>
<td>$668,361.00</td>
<td>$424,764.50</td>
<td>$243,596.50</td>
<td></td>
</tr>
<tr>
<td>Sep 02</td>
<td>$530,818.00</td>
<td>$424,764.50</td>
<td>$106,053.50</td>
<td></td>
</tr>
<tr>
<td>Oct 02</td>
<td>$568,429.00</td>
<td>$424,764.50</td>
<td>$143,664.50</td>
<td></td>
</tr>
<tr>
<td>Nov 02</td>
<td>$848,240.00</td>
<td>$424,764.50</td>
<td>$423,475.50</td>
<td></td>
</tr>
<tr>
<td>Dec 02</td>
<td>$1,445,116.00</td>
<td>$424,764.50</td>
<td>$1,020,351.50</td>
<td></td>
</tr>
</tbody>
</table>

**Advanced options for metric conditionality**

The process of merging the report and metric filters is accomplished by embedding one filter in the other or by embedding both filters in a new, empty filter. The Advanced options for conditionality allow you to select how the report filter and metric filter interact, as described below:

- **Merge report filter into metric**: is the default setting. The report filter criteria is applied to the data first. Then the metric filter is applied to the results of the first evaluation.

- **Merge metric condition into report**: evaluates the metric filter first, then applies the report filter to those results.

- **Merge into new**: intersects the metric and report filter.

These options are relevant only if at least one filter is dynamic, meaning that the filter results depend on when it is executed. For example, filtering on “country=US” always yields the same results. However, filtering on “country where revenue is greater than $1000” can return different results if the data is for 2002 only, 2003 only, or both years combined.
For example, you want to identify the bottom ten items in terms of revenue but you have placed an additional qualification on this: only those items where sales are greater than $30. You can solve this problem using the embedding options of metric conditionality.

The first qualification, the bottom ten items in terms of revenue, is contained within the metric, and the second, items with a value greater than $30, is in the report filter. By changing the embedding methods to control the interaction of these two filters, you alter the outcome.

- **Merge into New**: Merge into New intersects the metric filter and the report filter. In the example above, the result set includes only those items that were in the bottom 10 in terms of sales and had sales greater than $30. The report returns 10 rows of data.

- **Merge Report Filter into Metric**: This is the default setting. First, the report filter criterion is fulfilled and then the metric filter is applied to that result set. With this option chosen, first all the items having sales greater than $30 are found. Of these items, the bottom 10 sales are determined, so the report returns 10 rows again, although the data is different.

- **Merge Metric Filter into Report**: In this case, the metric filter is evaluated first, and based on this result set, the report filter is applied. That is, the bottom 10 sales are determined and then only those items with sales greater than $30 are returned on the report. The number of rows on the report will probably be less than 10, since some of the 10 items returned by the metric filter are likely to have sales less than $30.

The **Remove related report filter elements** check box also influences the interaction between the metric filter and the report filter. When it is selected, if the report filter contains a qualification based on an attribute related to an attribute in the metric filter qualification, the metric filter qualification takes precedence.
For example, the metric filter is set to New York only and the report filter to all western regions. The metric filter overwrites the report filter, so only New York is included in the report. If the check box is cleared, the results are the intersection of the filters. In this case, New York and West exclude each other, so the combined filter is empty.

By default, this option is selected since metric filters ignore report conditionality by default.

You can remember the difference between the advanced options available under the Level and the Condition components of a metric by remembering the following:

- The Add attributes in the filter to the level option determines whether the metric filter is applied to the metric calculation.

- The Remove related report filter elements option defines whether the report filter is applied to the metric calculation.

The **Remember option setting** check box allows you to save your settings as defaults.

### Transformation

Transformations are generally used for time-series analysis, for example, to compare values at different times such as this year versus last year, or month-to-date. For more information on transformations, see Chapter 17, *Transformations*.

A **transformation metric** is an otherwise simple metric that takes the properties of the transformation applied to it. For example, a metric calculates total revenue. Add a transformation for last year and the metric now calculates last year’s total revenue.

Any transformation can be included as part of the definition of a metric and multiple transformations can be applied to the same metric.
Definition of compound metrics

A compound metric is made by combining one or more other metrics using one or more mathematical operators. The other metrics may be simple, nested, or other compound metrics.

As noted in Distinguishing between simple and compound metrics, the most important distinction between simple and compound metrics is that compound metrics cannot have a level placed on the entire metric, although the level can be set separately on each of the expressions.

A compound metric does not have to perform an additional aggregation as a simple metric does. The two metrics can be calculated individually, instead. The results are used to compute the final result, as shown in the following figure.

For example, to calculate a profit margin, you can divide a profit metric by a revenue metric, as shown below:

\[
\left( \frac{\text{Profit Metric}}{\text{Revenue Metric}} \right)
\]

Note that this compound metric does not contain any levels; all of the level information is included in the separate metrics.
Smart metrics

The *smart metric* property of a compound metric allows you to change the default evaluation order of the metric. For example, the following report contains information about sales. If you display totals without allowing smart metrics, the totals are incorrect.

<table>
<thead>
<tr>
<th>Year</th>
<th>Total Sales</th>
<th>Discount Sales</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>200</td>
<td>50</td>
<td>25%</td>
</tr>
<tr>
<td>2003</td>
<td>100</td>
<td>50</td>
<td>50%</td>
</tr>
<tr>
<td>Total</td>
<td>300</td>
<td>100</td>
<td>75%</td>
</tr>
</tbody>
</table>

The Ratio column is summed to achieve the total shown. To calculate the correct answer, the total should be based on the totals in the other Total Sales and Discount Sales columns instead. Once smart metrics are allowed, the report looks like the following:

<table>
<thead>
<tr>
<th>Year</th>
<th>Total Sales</th>
<th>Discount Sales</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>200</td>
<td>50</td>
<td>25%</td>
</tr>
<tr>
<td>2003</td>
<td>100</td>
<td>50</td>
<td>50%</td>
</tr>
<tr>
<td>Total</td>
<td>300</td>
<td>100</td>
<td>33.3%</td>
</tr>
</tbody>
</table>

In short, smart metrics allow you to change the default evaluation order of a compound metric. Smart metrics calculate subtotals on the individual elements of the compound metric. For example, a smart metric uses the formula \(\text{Sum(Metric1)}/\text{Sum(Metric2)}\) rather than \(\text{Sum(Metric1/Metric2)}\).

To toggle smart metrics on and off, use the check box at the bottom of the Subtotals/Aggregation tab in the Metric Editor.
In the past, contribution metrics could not be smart metrics or totals would not work. For example, in the following MicroStrategy 7.1 report, the total for Contribution to Quarter would be as 25% if smart totals were turned on. This number would be calculated as the sum of all the quarterly contributions (100) divided by the sum of all the monthly contributions (400). If smart totals were turned off, the total would be correct at 100%.

<table>
<thead>
<tr>
<th>Month</th>
<th>Items Sold</th>
<th>Contribution to Month</th>
<th>Contribution to Quarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>30</td>
<td>100%</td>
<td>30%</td>
</tr>
<tr>
<td>February</td>
<td>50</td>
<td>100%</td>
<td>50%</td>
</tr>
<tr>
<td>March</td>
<td>20</td>
<td>100%</td>
<td>20%</td>
</tr>
</tbody>
</table>

MicroStrategy now offers dimensionally-aware subtotals, so the right answer is provided regardless of the smart metric setting. The only limitation for dimensional smart metrics is that subtotal values cannot be aggregated to a level higher than the metric itself, as illustrated below. The Items Sold and Contribution metrics cannot be rolled up to the year level. Turn off smart metrics in this case to achieve these results.

<table>
<thead>
<tr>
<th>Month</th>
<th>Items Sold</th>
<th>Contribution to Quarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>February</td>
<td>50</td>
<td>50%</td>
</tr>
<tr>
<td>March</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Quarter Subtotal</td>
<td>100</td>
<td>100%</td>
</tr>
<tr>
<td>Year Subtotal</td>
<td>----------</td>
<td>------------------------</td>
</tr>
</tbody>
</table>
Evaluation order

The order in which data is calculated has a bearing on the results to be displayed. By using evaluation order settings, you can control the order in which consolidations, smart metrics, report limits, and subtotals are calculated and resolved for a given report. It is important to think about evaluation order when creating compound metrics. For more information on evaluation order, see Chapter 2, *Reports*.

Metric aggregation and subtotals

Aggregation and subtotals allow you to control how metrics are further computed or rolled up. The functions that are used for both types of operations are shown below.

<table>
<thead>
<tr>
<th>Aggregation type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Count[count] number of input values</td>
</tr>
<tr>
<td>Average</td>
<td>Avg[average] sum of input values divided by number of input values</td>
</tr>
<tr>
<td>Minimum</td>
<td>Min[minimum] smallest input value</td>
</tr>
<tr>
<td>Maximum</td>
<td>Max[maximum] largest input value</td>
</tr>
<tr>
<td>Product</td>
<td>Product[product] all input values multiplied together</td>
</tr>
<tr>
<td>Median</td>
<td>Median[median] middle value when all values are sorted</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode[mode] most frequently found input value</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>Stdev[standard deviation] distribution of input values</td>
</tr>
</tbody>
</table>
These functions reflect only those most frequently used for further aggregation of metrics or for evaluating metric subtotals. The Analytical Engine also can handle a large number of statistical, mathematical, financial, date-and-time, string, and OLAP functions, from simple to highly complex. To reach these functions, use the Object Browser, and select Functions and Operators.

You can also create user-defined subtotals, which allow you to develop your own subtotals, using aggregation or nested functions, constants, and combinations of these objects. For more information, see the What are user-defined subtotals? section in Chapter 2, Reports. A user-defined subtotal is indistinguishable from the standard predefined subtotals functions such as total or count.

### Subtotals

In the context of metrics, subtotals permit computation and display of quantified data, gathered by MicroStrategy, along attribute groupings that you can specify dynamically for a report. The behavior of subtotal aggregations is based on the types of data included in the metric to which the subtotal is applied.

The subtotals function allows you to determine which subtotals are available for that metric. When a report containing that metric is run, a user can choose which of these subtotals to display. You can also choose to completely block totaling on the metric.

<table>
<thead>
<tr>
<th>Aggregation type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance</td>
<td>$\text{Var} [\text{variance}]$ square of the distribution of input values</td>
</tr>
<tr>
<td>Geometric mean</td>
<td>$\text{Geomean} [\text{geometric mean}]$ square root of the product of input values</td>
</tr>
</tbody>
</table>
Dynamic aggregation

Dynamic aggregation by the Analytical Engine occurs when objects are moved between the grid and the Report Objects in the Report Editor. The metric values roll up to the new level of the grid. Dynamic aggregation happens on the fly, in memory. For an example of dynamic aggregation, see the Dynamic aggregation section in Chapter 2, Reports.

The dynamic aggregation setting allows you to specify what function to use when the Analytical Engine aggregates the metric. The default setting allows the Analytical Engine to choose the function for dynamic aggregation, according to these rules:

- If the metric is a compound metric, sum is used as the aggregation function.
- If the metric is a sum, maximum, minimum, or other expression that can be recalculated, the metric's expression is used.
- If the metric cannot be recalculated, as with an average or count distinct, the metric values are replaced with dashes to signify that the metric cannot be calculated at this level.

The ability to roll up data in memory is useful for quick report interaction and analysis. However, not all metrics can be rolled up with an additional aggregation function. Instead, if the data is required at the higher level, it first must be recalculated from the detail data available only in the data warehouse. For example, a metric is defined as Avg(Revenue){~+} with dynamic aggregation set to default. When an attribute is removed from a report that contains this metric, the revenue average values are replaced with dashes, signifying that the metric cannot be calculated at this level. For more information, see Exceptions to dynamic aggregation in Chapter 2, Reports.
Join specification

Setting a join specification allows you to place conditions on the data selected for display in a report. You can apply an inner or outer join, which are described in more detail below. In short, an inner join includes only the data common to all the elements in the join, whether tables or metrics. An outer join includes all of the data in all of the elements. You can set joins at the metric and report levels:

- metric: how the metric is joined to other metrics
- report: how metrics are joined together in the report; overrides metric join settings for that report only

Setting the metric join type at the report level (that is, using the Report Data Options menu option in the Report Editor) affects only the results for the report being modified. To set the join type globally, specify it at the metric level, using the Metric Join Type option on the Tools menu of the Metric Editor. A metric join type that is set globally affects the results for all reports using that metric.

For compound metrics, you can also set the join type at the formula level. This controls how the expressions or metrics within the compound metric are joined.

Inner joins versus outer joins

In short, an **inner join** includes only data that is common across all component of the join. An **outer join** includes data that applies to all components. The following examples explain this in more detail.

By default, inner joins are generated against all metrics in a report. The resulting report contains only those rows that have data returned for all the metrics. For example, review the data in the following table.

<table>
<thead>
<tr>
<th>Region</th>
<th>Sales Information?</th>
<th>Budget Information?</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>South</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
A report is created containing Sales and Budget metrics by region. The default inner join is not changed. Since the North region does not have any budget data, it is not displayed on the report. Similarly, sales data has not been tracked for the West, so it is also omitted from the report. The resulting report, with an inner join between metrics, displays only those regions that have both sales and budget information. It looks like the following.

<table>
<thead>
<tr>
<th>Region</th>
<th>Sales Information?</th>
<th>Budget Information?</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>West</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

However, you may want to display all of the rows in the report, whether data exists for all the metrics in the report. An outer join on both metrics results in the following report, where North and West are listed even though they have no data for one of the metrics.

<table>
<thead>
<tr>
<th>Region</th>
<th>Sales</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>South</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td>East</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>North</td>
<td></td>
<td></td>
</tr>
<tr>
<td>South</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td>East</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>West</td>
<td></td>
<td>300</td>
</tr>
</tbody>
</table>
Finally, you can specify different joins for each of the metrics on a report. If the Sales metric is defined with an outer join and the Budget metric with an inner join, only those rows with information in sales are displayed. The following report is created.

<table>
<thead>
<tr>
<th>Region</th>
<th>Sales</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>South</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td>East</td>
<td>100</td>
<td>400</td>
</tr>
</tbody>
</table>

West is not displayed because it does not contain sales information. It is irrelevant if data exists for the Budget metric.

**Formula join type for compound metrics**

A compound metric contains multiple expressions or metrics. You can define how these elements are joined, using the Metric Formula dialog box. This dialog box is accessed from the Advanced Settings option in the Metric Editor Tools menu. The join types for a metric base formula are as follows:

- A default join is a join that is defined in each element.
- An inner join includes only data that is common across all elements.
- An outer join includes data that apply to every metric in a report.
Joins between metrics

Setting the metric join type allows you to define the default action for joining the metric to other metrics. This setting is available from the Metric Join Type option in the Metric Editor Tools menu. The metric join types are as follows:

- Default uses the default value.
- Inner includes only information contained by all the elements.
- Outer keeps all the information in all the elements.

The Metric Join Type option is a shortcut to the Metric Join Type VLDB property located under Advanced Options in the same menu.

Metric-specific VLDB properties

There are other VLDB properties besides joins that affect metrics. **VLDB properties** allow MicroStrategy products to exploit the unique optimizations offered by different databases. These settings affect how MicroStrategy Intelligence Server manages joins, metric calculations, and query optimizations, among others. VLDB properties are available at multiple levels, so that SQL generated for one report can be manipulated separately from the SQL generated for a different report. The hierarchy, or order of precedence, for VLDB properties is outlined in the following figure.
The arrows depict the override authority of the levels, with the report level having the greatest authority. For example, if a VLDB property is set one way for a report, and the same property is set differently for a metric included on the report, the report property takes precedence.

Although there are ten VLDB properties that pertain specifically to metrics, there are only six metric VLDB properties available at the individual metric level. You can set additional metric VLDB properties at other levels, such as the report and project levels. Refer to the MicroStrategy System Administration Guide for a detailed description of these properties.

Metric-specific VLDB properties that can be set at the metric level include

- Integer Constant in Metric
- Metric Join Type
- Null Check
- Zero Check
- Null Checking for Analytical Engine
- Subtotal Dimensionality Aware

To set these properties, select Advanced Settings from the Tools menu in the Metric Editor. Then choose VLDB Properties to access the VLDB Properties (Metric) dialog box. The last two settings in the list above are contained under the Analytical Engine folder, while the others are found in the Metrics folder.

**Metric VLDB properties**

**Integer Constant in Metric**

This setting determines whether to add a “.0” after the integer. The options for this property are as follows:

- Add “.0” to integer constants in metric expressions.
- Do not add “.0” to integer constants in metric expressions.
- Use the default inherited value.

**Metric Join Type**

This property sets the type of join used in the metric. The options are

- Inner join, which includes only data that is common across all elements
- Outer join, which includes data that apply to every metric in a report
- The default inherited value

For more information on join types, see *Join specification*. 
Null Check

The Null Check property indicates how to handle arithmetic operations with null values. The options for this property are as follows:

- Do nothing, which means that the database rather than the Analytical Engine handles division by a null value.
- Check in all queries.
- Check in temporary table join only.
- Use the default inherited value, from the DBMS level.

Zero Check

The Zero Check property indicates how to handle division by zero or when to check for zeros in the denominator during division operations. The options for this property are as follows:

- Do nothing, which means that the database rather than the Analytical Engine handles division by a zero.
- Check in all queries.
- Check in temporary table join only.
- Use the default inherited value, from the DBMS level.

Analytical Engine VLDB properties for metrics

Null Checking for Analytical Engine

This setting determines whether a null value is interpreted as zero when the Analytical Engine performs calculations. The options for this property are as follows:

- False, which means that null values are not altered
- True, which means the Analytical Engine converts null values to zeros
• The default inherited value

You can also set replacement text for nulls at the report level. For more information, see Chapter 2, *Reports*.

**Subtotal Dimensionality Aware**

The Subtotal Dimensionality Aware setting enables subtotaling based on the dimensionality of a metric. How it works depends on another VLDB property, Query Population Dimensionality Aware, which handles backwards compatibility with MicroStrategy 7.1. These settings work together as illustrated in the following table.

<table>
<thead>
<tr>
<th>If Query Population Is:</th>
<th>Then Subtotal Is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>TRUE or FALSE</td>
</tr>
<tr>
<td>FALSE</td>
<td>Ignored (meaning that subtotaling is never aware of dimensionality)</td>
</tr>
</tbody>
</table>

The default setting for the Subtotal Dimensionality Aware property is TRUE, so subtotals depend on the metric’s dimensionality. If you must subtotal without using the dimensionality of the metric, set this property to FALSE.

The options for this property are:

• FALSE, which means that subtotals do not take into account the dimensionality of the metric

• TRUE which means that subtotaling is aware of metric dimensionality

• The default inherited value
For example, the Quarterly Revenue metric is defined as 
\( \text{Sum(Revenue)} \) Dimensionality = Quarter, and the 
Yearly Revenue metric is defined as \( \text{Sum(Revenue)} \) 
Dimensionality = Year.

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Quarterly Revenue</th>
<th>Yearly Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>100</td>
<td>600</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>200</td>
<td>600</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>100</td>
<td>600</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>200</td>
<td>600</td>
</tr>
</tbody>
</table>

If Subtotal Dimensionality Aware is set to FALSE, the 
quarterly subtotal is calculated as 600, that is, a total of the 
Quarterly Revenue values. The yearly subtotal is calculated as 
2400, the total of the Yearly Revenue values. This is the way 
MicroStrategy 7.1 calculated the subtotal.

If Subtotal Dimensionality Aware is set to TRUE, the 
quarterly subtotal is still 600. MicroStrategy is aware of the 
dimensionality of the Yearly Revenue, so rather than simply 
adding the column values, it calculates the total as 600.

**Metric column alias**

*Column aliases* allow you to modify the column names of 
existing metrics for use in temporary tables without affecting 
the original column name. For example, a temporary table 
may be created in the SQL used to generate a report. If you 
alias the column, you can identify it easily. Column aliases 
can be set from the Advanced Settings option under the Tools 
menu of the Metric Editor. You can also set the data type and 
byte length for the metric. The MicroStrategy data types are 
Big Decimal, Binary, Char, Date, Decimal, Double, Float, 
Integer, LongVarBin, Long.VarChar, Numeric, Real, Time, 
Timestamp, Unsigned, VarBin, and VarChar.

There are several drawbacks to using Big Decimal data 
type for metric values. For more information, see 
Appendix B, *Data types*. 

© 2005 MicroStrategy, Inc.
Formatting metrics

You can format metrics to control how they are displayed on reports, that is, to specify the format for a particular metric, regardless of the report the metric is on. This formatting, which can be applied to headers and data values, is performed in the Metric Editor or through a find and replace operation. Metric level formatting is overwritten by axis and all metrics formatting, which occurs on the report level. Those formatting layers must be set to default to allow the metric level formatting to display. For more information on formatting layers, see Formatting in Chapter 2, Reports.

The following tables contain format-related specifics for

- number display codes
- symbols
- colors

Report-formatting options described in this section are available in the Metric Editor, the Report Editor/Viewer, and the Find and Replace dialog box. To access these options in the Metric Editor, select Tools, then Formatting. In the Report Editor/Viewer, it is available from the Format menu. To open the Find and Replace dialog box, log in to a project with administrative or desktop designer privileges and select Tools, then Find and Replace...

You can use this information to determine how the built-in formats will display data or to create custom number formatting if the built-in styles do not meet your needs.
# Number display codes

Formatting codes are used to select formats for data values of metrics in a grid report. The table that follows shows format codes for the various types of available value displays, specifying differences between positive, negative, and decimal values.

<table>
<thead>
<tr>
<th>Number Category</th>
<th>Format Code</th>
<th>Positive</th>
<th>Negative</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>General</td>
<td>3</td>
<td>-3</td>
<td>.3</td>
</tr>
<tr>
<td>Number</td>
<td>0</td>
<td>3</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>3.00</td>
<td>-3.00</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>#,##0</td>
<td>3</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>#,##0.00</td>
<td>3.00</td>
<td>-3.00</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>#,##0;(#,##0)</td>
<td>3</td>
<td>(3)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>#,##0.00;(#,##0.00)</td>
<td>3.00</td>
<td>(3.00)</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>#,##0:<a href="#,##0">RED</a></td>
<td>3</td>
<td>(3)[in red]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>#,##0.00:<a href="#,##0.00">RED</a></td>
<td>3.00</td>
<td>(3.00)[in red]</td>
<td>0.30</td>
</tr>
<tr>
<td>Currency</td>
<td>$#,##0;($#,##0)</td>
<td>$3</td>
<td>($3)</td>
<td>$0</td>
</tr>
<tr>
<td></td>
<td>$#,##0:<a href="$#,##0">RED</a></td>
<td>$3</td>
<td>($3)[in red]</td>
<td>$0</td>
</tr>
<tr>
<td></td>
<td>$#,##0.00;($#,##0)</td>
<td>$3.00</td>
<td>($3.00)</td>
<td>$0.30</td>
</tr>
<tr>
<td></td>
<td>$#,##0.00:<a href="$#,##0.00">RED</a></td>
<td>$3.00</td>
<td>($3.00)[in red]</td>
<td>$0.30</td>
</tr>
<tr>
<td>Percent</td>
<td>0%</td>
<td>300%</td>
<td>-300%</td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>0.0%</td>
<td>300.0%</td>
<td>-300.0</td>
<td>30.0%</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>300.00%</td>
<td>-300.00</td>
<td>30.00</td>
</tr>
<tr>
<td>Fraction</td>
<td>#/?</td>
<td>3</td>
<td>-3</td>
<td>2/7</td>
</tr>
<tr>
<td></td>
<td>#/?/??</td>
<td>3</td>
<td>-3</td>
<td>3/10</td>
</tr>
<tr>
<td>Scientific</td>
<td>0.00E+00</td>
<td>3.00E+00</td>
<td>-3.00E+00</td>
<td>3.00E-01</td>
</tr>
<tr>
<td></td>
<td>#0.0E+0</td>
<td>3.0E+0</td>
<td>-3.0E+0</td>
<td>3.0E-1</td>
</tr>
</tbody>
</table>

© 2005 MicroStrategy, Inc.
Symbols and their functions

The following table shows the symbols used for display formatting and the function associated with each.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Displays a number in general format.</td>
</tr>
</tbody>
</table>
| 0 | Digit placeholder. Conditions are as follows:  
  • If a number contains fewer digits than this placeholder, the number is padded with zeros.  
  • If a number contains more digits to the right of the decimal point than this placeholder, the decimal component of the number is rounded off to fit the number of digits that this placeholder can accommodate.  
  • If a number contains more digits to the left of the decimal point than this placeholder, the extra digits are retained. |
| # | Digit placeholder. The function is very similar to that of 0 (see above), but there is no zero padding when the number contains fewer digits than this placeholder. |
| ? | Digital placeholder. The function is very similar to that of 0 (see above), but padding uses spacing in place of zeros for padding. |
| . (period) | Decimal point. Determines the number of digits displayed on either side of the decimal point. Conditions are as follows:  
  • If the format contains only #s to the left of the decimal point, values less than one begin with a decimal point.  
  • If the format contains zeros to the left of the decimal point, values less than one begin with a zero. |
| % | Displays a number as a percent value. The original number is multiplied by 100 then a “%” is appended. |
| , (comma) | Thousands separator. Conditions are as follows:  
  • If the format contains commas separated by placeholders # or 0, the number is displayed with a comma for every three integral places.  
  • A comma following a # or 0 placeholder implies an increment by a factor of 1000. For example, the number 10,000 would be represented as 10. |
### Formatting metrics

- **E- E+ e- e+**
  Displays a number in scientific notation. Conditions are as follows:
  - If the format includes scientific notation symbols to the left of a # or 0 placeholder, the number is displayed in scientific notation, with either E or e added.
  - The number of placeholders (# or 0), either to the right or to the left of the decimal point, determines the value of the exponent.
  - E- and e- place a “-” (minus) sign next to a negative exponent.
  - E+ and e+ place a “-” (minus) sign next to a negative exponent, and a “+” (plus) sign next to a positive exponent.

- **$ - + / (): space**
  Displays the character. To display a character not on this list, either precede that character with a backslash (\) or enclose it in double quotes. Note that the backslash is also used to format fractions.

- **\**
  Displays the character that follows. The backslash symbol itself is not displayed.

- **"(text)"**
  Displays enclosed text.

- **@**
  Text placeholder, where text replaces @.

- *** **
  Repeats the character that follows across the width of the column. A format section can have only one asterisk.

- **_ (underscore)**
  Skips the width of the character that follows. For example, to align negative numbers surrounded by parentheses with positive numbers, enter _ for the positive numbers to skip the width of each parenthesis.

- **m**
  Month number. Displays the month as digits without leading zeros, such as 1. Can also represent minutes when used with h or hh formats.

- **mm**
  Month number. Displays the month as digits with leading zeros, as in 01. Can also represent minutes when used with the h or hh formats.

- **mmm**
  Month abbreviation, such as Jan.

- **mmmm**
  Month name, such as January.

- **d**
  Day number. Displays the day as digits with no leading zero, such as 1.

- **dd**
  Day number. Displays the day as digits with leading zeros, as in 01.

- **ddd**
  Day abbreviation, such as Sun.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddddd</td>
<td>Day name, such as Sunday.</td>
</tr>
<tr>
<td>yy</td>
<td>Year number. Displays the year as a two-digit number, such as 00.</td>
</tr>
<tr>
<td>yyyy</td>
<td>Year number. Displays the year as a four-digit number, such as 2002.</td>
</tr>
<tr>
<td>g</td>
<td>If you are using a Japanese locale, displays the Latin letter for an era.</td>
</tr>
<tr>
<td>gg</td>
<td>If you are using a Japanese locale, displays the first character of an era name.</td>
</tr>
<tr>
<td>ggg</td>
<td>If you are using a Japanese locale, displays the full era name.</td>
</tr>
<tr>
<td>e</td>
<td>If you are using a Japanese locale, displays the full era year.</td>
</tr>
<tr>
<td>ee</td>
<td>If you are using a Japanese locale, displays the full era year with a leading zero if the year is less than ten.</td>
</tr>
<tr>
<td>h</td>
<td>Hour number. Displays the hour as a number without leading zeros, such as 1. If the format contains an AM or PM format, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock.</td>
</tr>
<tr>
<td>hh</td>
<td>Hour number. Displays the hour as a number with leading zeros, as in 01. If the format contains an AM or PM format, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock.</td>
</tr>
<tr>
<td>m</td>
<td>Minute number. Displays the minute as a number without leading zeros, such as 0. The m format must appear immediately after the h or hh symbol otherwise it is interpreted as month.</td>
</tr>
<tr>
<td>mm</td>
<td>Minute number. Displays the minute as a number with leading zeros, such as 00. The mm format must appear immediately after the h or hh symbol otherwise it is interpreted as month.</td>
</tr>
<tr>
<td>s</td>
<td>Second number. Displays the second as a number without leading zeros, such as 0.</td>
</tr>
<tr>
<td>ss</td>
<td>Second number. Displays the second as a number with leading zeros, such as 00.</td>
</tr>
<tr>
<td>AM/PM</td>
<td>12-hour time. Displays time using a 12-hour clock. Displays AM, am, A, or a for time between midnight and noon; displays PM, pm, P, or p for time from noon until midnight.</td>
</tr>
<tr>
<td>[h]</td>
<td>Total number of hours.</td>
</tr>
</tbody>
</table>
Colors

Available colors for metric formatting include

- black
- blue
- cyan
- green
- magenta
- red
- white
- yellow

Creating metrics in the Report Editor

You can create metrics in the Metric Editor, the Report Editor, or the Command Manager. This chapter as a whole describes the concepts behind metric creation in all of these components.

This section discusses metrics created on the fly in the Report Editor, which are called derived metrics. Derived metrics allow you to create calculations based on the data already available on a report. A particular group of derived metrics, shortcut metrics, uses pre-built formulas.
One category of shortcut metric, the transformation metric, is not a derived metric. Unlike other shortcut metrics, they must be calculated in SQL rather than in the Analytical Engine.

### Derived metrics

A derived metric is developed in the context of a report, allowing you to create calculations based on the data already available in the report. For example, a report contains Revenue and Cost metrics. To calculate profit on the fly, create a derived metric in the Report Editor with the following definition:

\[
([\text{Revenue Metric}] - [\text{Cost Metric}])
\]

Derived metrics are always evaluated by the Analytical Engine. That is, a derived metric performs math between metrics (column math) in report data after it has been returned from the data warehouse. The definition of a derived metric is visible in the formula bar of the Report Editor. Formulas of non-derived metrics are not visible.

In their structure, derived metrics:

- may include one or more metric functions
- are based on the attributes and metrics that currently exist in the report
- may be simple or compound, and therefore will inherit the structure of whichever type you use.

For example, if you have a report with Calling Center, Unit Price, and Units Sold, you could create the following derived metrics:

\[
[\text{Unit Price}] \times [\text{Units Sold}]
\]

\[
\text{Max (Units Sold)} \{\}
\]

Derived metrics are local to the report in which they are created. A derived metric created in a report is not a new metric and is not available to other reports.
Derived metrics are usually compound metrics based on the metrics that already exist in the report. The expressions of these metrics may include simple functions and OLAP functions, as shown in the examples below:

\[
\text{Rank}([\text{Units Sold}])
\]
\[
\text{RunningSum<SortBy=(Month)>}(\text{Revenue})
\]

**Aggregation and Dimensionality**

When you create a derived metric, you cannot access all of the advanced metric functionality, such as transformations and conditions. However, you can employ levels, or dimensionality.

If you define a derived metric using an aggregate function such as sum or maximum, you must also specify the level. The default is to group by nothing, as in \( \text{SUM(Metric1)} \{\} \). You can list specific report attributes, such as \( \text{SUM(Metric1)}\{\text{Region, Employee}\} \) to force a specific level of aggregation. You cannot use other advanced metric capabilities. For example, filtering options such as Absolute and Standard filtering are not supported.

When aggregating data within a derived metric, the metric values are first calculated for the metric at the grid level and then the aggregate function in the derived metric is applied from the grid level to the level specified in the derived metric. For example, consider a report with Category, Subcategory, and Item in the report objects and Category, Subcategory in the grid. With a derived metric Category Sales defined as \( \text{Max(Revenue)} \{\text{Category}\} \), the Revenue metric will be calculated at the Category and Subcategory level and the Category Sales metric will be calculated as the maximum of those Revenue values at the Category level.
Inserting Embedded Metrics

While you can create a non-derived metric such as \( \text{SUM} (\text{Fact}) \{~+\} \) directly in a report, it becomes an embedded metric and exists only in the context of the report. Its definition can be edited or viewed, just as with a regular object. Metric formulas that contain facts or attribute forms will result in embedded metrics and will not be evaluated by the Analytical Engine like derived metrics.

Shortcut metrics

Shortcut metrics are based on metrics already included in a report and provide a quick way to add new metrics to the report. They are available when you right-click on a metric column or metric header and are based on the selected metric.

Shortcut metrics belong to one of the following categories:

- percent-to-total metrics
- transformation metrics
- rank metrics

All shortcut metrics are derived metrics except for the transformation metrics, which must be calculated in SQL. All the shortcut types are described in more detail below.
Percent-to-total metrics

Percent-to-total metrics display the percent, in relation to a selected total of each item affected by the metric. The total can be by column, by row, by page, for each value of the attribute, or the grand total. Associated calculation levels are shown in the following table.

<table>
<thead>
<tr>
<th>Percent-to-Total</th>
<th>Calculation Level of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Over Rows</td>
<td>All the attributes in the column and page-by axis; displays values in each row of the report as percents of a row total.</td>
</tr>
<tr>
<td>Over Columns</td>
<td>All the attributes in the row and page-by axis; displays values in each column of the report as percents of a column total.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Use only if a column contains an attribute.</td>
</tr>
<tr>
<td>Page Total</td>
<td>All the attributes in the page axis; displays all values on a page as percents of that page’s total.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This calculation is only applicable to reports with a Page-by function.</td>
</tr>
<tr>
<td>Grand Total</td>
<td>No level specified. Aggregates across all the data in the report; displays all values in a report as percents of the grand total for that report.</td>
</tr>
<tr>
<td>Total for Each</td>
<td>The selected attributes; displays all values pertaining to a given attribute element as percents of the total accumulated for that attribute element.</td>
</tr>
</tbody>
</table>
An example of a percent-to-total metric begins with the following report.

<table>
<thead>
<tr>
<th>Gender</th>
<th>Age Range</th>
<th>&gt; 100K</th>
<th>50K-100K</th>
<th>&lt; 50K</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>&gt; 50</td>
<td>200</td>
<td>400</td>
<td>80</td>
<td>680</td>
</tr>
<tr>
<td>Female</td>
<td>40-50</td>
<td>50</td>
<td>300</td>
<td>160</td>
<td>510</td>
</tr>
<tr>
<td>Female</td>
<td>30-40</td>
<td>40</td>
<td>180</td>
<td>240</td>
<td>460</td>
</tr>
<tr>
<td>Female</td>
<td>&lt; 30</td>
<td>10</td>
<td>120</td>
<td>320</td>
<td>460</td>
</tr>
<tr>
<td>Female</td>
<td>TOTAL</td>
<td>300</td>
<td>1000</td>
<td>800</td>
<td>2100</td>
</tr>
</tbody>
</table>

If Percent-to-Column Total is selected as the metric type, the information is displayed as follows.

<table>
<thead>
<tr>
<th>Gender</th>
<th>Age Range</th>
<th>&gt; 100K</th>
<th>50K-100K</th>
<th>&lt; 50K</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>&gt; 50</td>
<td>67%</td>
<td>40%</td>
<td>10%</td>
<td>32%</td>
</tr>
<tr>
<td>Female</td>
<td>40-50</td>
<td>17%</td>
<td>30%</td>
<td>20%</td>
<td>24%</td>
</tr>
<tr>
<td>Female</td>
<td>30-40</td>
<td>13%</td>
<td>18%</td>
<td>30%</td>
<td>22%</td>
</tr>
<tr>
<td>Female</td>
<td>&lt; 30</td>
<td>3%</td>
<td>12%</td>
<td>40%</td>
<td>21%</td>
</tr>
<tr>
<td>Female</td>
<td>TOTAL</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

The following conditions apply to percent-to-total metrics.

- Row and column percent totals refer to the uppermost and extreme-left positions, respectively.
- Page percent totals affect all attributes on a page.
- “Percent to All -> A1”, where A1 is an attribute, indicates that the calculation is performed across all elements of that attribute. An example is percent-to-all stores.
- If a report does not contain attributes at a given percent-to-total level, the level is unavailable for that report.
- In some cases, two or more percent-to-total calculations at different logical levels yield the same result. For example, Percent-to-Page Total data can be the same as Percent-to-Grand Total data in a single-page report.
- The level of a percent-to-total metric remains constant once the metric has been calculated; subsequent manipulation does not affect it.
Transformation metrics

Transformation metrics apply offset values, such as “four months ago,” to the selected attribute. For the MicroStrategy Tutorial, the offset values for the shortcuts are

- two weeks ago
- last month
- month to date
- last year
- previous
- last quarter

These transformations are included as examples in the Tutorial. In your project, the options are the transformations that have been created for the project.

For each of these values, you can select what to calculate:

- normal to show unit figures for both the current values and the corresponding ones for the interval selected
- variance to display the difference between the current values and the corresponding ones for the interval selected, for example, Revenue - (Last Year’s (Revenue))
- variance percentage to calculate the difference, expressed as a percentage, between the current values and the corresponding ones for the interval selected, for example, Revenue - (Last Year’s (Revenue))/(Last Year’s (Revenue))

Rank metrics

Rank metrics apply a ranking number to the metric values for a given attribute. When selected, this shortcut metric provides break-by options for each attribute on the report.

By default, the rank derived metric performs an ascending sort. To change it to descending, edit the metric and replace <ASC=True> with <ASC=False>. 
Using a transformation metric affects the results of all subsequent executions of the report.

For more information on the rank function, see *Useful functions*.

## Useful functions

Some useful functions to use in metrics include:

- rank
- count
- running and moving sums and averages (or OLAP functions)
- N-tile
- first and last

All of the above except count and first/last are non-group functions.

### Rank

In rank functions, you specify the metric to be ranked and whether to rank in ascending or descending order. You can also specify whether to break by an attribute.

The level of the rank depends on the level in the report. For example, the following metric combined with customer on a report shows the highest revenue customer as number one.

```
Rank (Revenue) <Ascending+False>
```

Adding a parameter to break by year displays the customer generating the highest revenue in each year as number one.
Count

The count function is usually used with an attribute, although a fact can also be counted. You can specify whether to count from a fact or a lookup table and whether to count distinct occurrences of the target.

For example, a metric can count the number of customers in a specified region, based on the lookup table. Another metric calculates the number of customers who generated revenue in a particular month. This calculation is based on a fact table containing the revenue fact.

Running and moving sums and averages

These functions include

- moving average
- moving sum
- running average
- running sum

All these functions are similar and are called OLAP functions in the Metric Editor. The running sum function uses a metric as input and calculates a cumulative total of values based on a sort order specified in the metric definition. The sort order is not based on the report level. For example, a report with dates, revenue, and month-to-date revenue is needed. The month-to-date revenue is defined as

\[
\text{RunningSum(Revenue) \text{ <Sort Ascending by Date>.}}
\]

For input, the moving sum and average require a metric and a window size, that is, a date range.
N-tile

The N-tile function, which is also referred to as segmentation, sets up numbers of groups, or tiles, for a metric. Examples of required parameters are the number of groups and whether they should be sorted in ascending or descending order.

An example of an N-tile function in use is displaying what items are in the top 25% of sales, the next 25%, and so on. Use the N-tile function with the Revenue metric. Because the results are in quartiles (four groups of 25 each), the number of tiles is four.

First and Last

The First and Last functions provide the ability to use sort-by inside aggregation functions, that is, functions where the value depends on the sort order. First returns the First value in a sorted set of values, while Last returns the last value. You can define the sort attributes in the function parameters.

For example, an inventory report lists the on-hand supply for each day. The report subtotals are the last day’s inventory. Creating a user-defined subtotal that uses the Last function provides the needed last day inventory subtotal. If the sort parameters of the function are not set to sort by Day, the function may not provide the correct answer.

For a sample scenario using the First function, see User-defined subtotal example (First function) in Chapter 2, Reports. For more details on the functions themselves, see the MicroStrategy Analytical Engine Functions Reference.

Creating metrics in the Command Manager

If you understand the various building blocks of metrics, you can create a metric without using the Metric Editor. Instead, you work in the Command Manager.
The syntax of a metric is:

```
Function<parameter>{arguments}{level}<filter> |transformation|
```

The syntax of each of these components is discussed below. Only a brief overview of the function of the component is provided. For details, see the appropriate section earlier in this chapter.

## Operators and functions

An operator enables calculation by providing the mathematical expression, such as addition or rank, for a given metric definition. Closely associated with the operator, delimiters show the structure of a metric definition by enclosing each metric component within recognizable symbols when displayed. Delimiters include those for

- object type ([ ])
- level (dimensionality) ({ })
- filter (< >)
- transformations ( ||)

Operators and delimiters are displayed in a metric definition as follows:

```
METRICNAME (Fact) {Metric level} <Filter>|Transformation|
```
Level

The level of a metric determines how the MicroStrategy Engine selects and assembles information for calculation. The following applies to the contents of the level statement in a metric definition:

- The statement is enclosed in brackets ( { } ).
- Statement components are separated by commas.
- If you are using a regional setting other than English, use the list separator that is defined in the regional settings of your machine. If you do not, errors can occur when you edit an object.
- Statement components are associated with special characters that determine grouping or filtering values.
- Modifiers that appear before an attribute name denote grouping values.
- Modifiers that appear after an attribute name denote filtering values.
- An attribute-grouping-filtering content combination is known as a level unit.

The figure shows the order of element placement within the level statement of a metric definition.

```
Sum (Fact) { ~ +, Region+ }
```

In the context of metric levels, the tilde ( ~ ) indicates the level given by the report. If no other information is present in the level portion of a metric’s definition, it signifies that the metric is aggregated at the report level.
Level filtering

When determining the calculations to be performed for a given metric, you can change how the WHERE clause in the SQL statement affects certain attributes in a report. Use level filtering to modify metric definitions in this way. The following table shows the options available when applying a filter to a level metric.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>+</td>
</tr>
<tr>
<td>Absolute</td>
<td>*</td>
</tr>
<tr>
<td>Ignore</td>
<td>%</td>
</tr>
<tr>
<td>Undefined</td>
<td>Empty</td>
</tr>
</tbody>
</table>

**Standard filtering**

Standard filtering does not change the filter applied to the report. For example, a metric is defined as:

\[
\text{Sum} \ (\text{Reg\_Sls\_Dlr}) \ \{ \sim + \}
\]

To specify Region as the aggregation level and leave the WHERE clause in the SQL statement intact, change the metric to the following:

\[
\text{Sum} \ (\text{Reg\_Sls\_Dlr}) \ \{ \sim +, \text{Region}+ \}
\]

**Absolute filtering**

Absolute filtering raises elements in the report filter to the level of the specified attribute. For example, a report contains a filter at the market level and uses this metric:

\[
\text{Sum} \ (\text{Reg\_Sls\_Dlr}) \ \{ \sim +, \text{Region}\ast \}
\]

Only regions having children included in the report filter are included in the calculation. If the report has Region as its filter, the metric uses that filter without modification.
Ignore filtering

Ignore filtering is used when a given attribute is not included in metric calculations. For example, a metric is defined as:

\[ \text{Sum (Reg_Sls_Dlr) \{ ~ +, Region\%} \]

If the report has a filter on Market, which is a child of Region, that filter would not be part of the calculation.

Undefined filtering

Undefined filtering can be summarized as unspecified—the filtering behavior for the target is not determined by this component. Instead, the target and group components of this level unit define the filter. For example, a metric is defined as:

\[ \text{Sum (Reg_Sls_Dlr) \{ ~ +, Region+, Market \}} \]

The WHERE clause in the SQL statement is affected by the value for Region because Market is undefined.

Undefined filtering is referred to as None in the Metric Editor.

Level grouping

Within the level statement, grouping information determines the level at which the metric function aggregates data. Specifically, this level component allows you to select

- an aggregation level different from that specified by the report filter
- either the first or last entry in a range of values

The table shows how these options are displayed in the level statement.

<table>
<thead>
<tr>
<th>Grouping</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Empty</td>
</tr>
<tr>
<td>Beginning fact</td>
<td>&gt;</td>
</tr>
</tbody>
</table>
When an attribute is part of the metric level, calculations are aggregated at that attribute’s level by default, rather than at the report level. For example, a metric is defined as:

\[ \text{Sum (Reg_Sls_Dlr) \{ ~ +, Region+} \]

Region as part of the level statement implies that the aggregation occurs at this level, rather than at the report level of Store. The effect is equivalent to obtaining subtotals for Region and applying the data to all rows.

**Beginning fact and beginning lookup grouping**

When grouping by beginning values, whether from a fact table or a lookup table, aggregation occurs at a lower level than that indicated. The value used for the report is the first one found on the table. These options are only used with nonaggregatable metrics or facts.

Applying the beginning fact grouping option, the metric definition reads:

\[ \text{Sum (Reg_Sls_Dlr) \{ ~ +, <|Store+} \]

Applying the beginning lookup grouping option calls for the use of lookup-table information, so that the first value used for display is an attribute element, not a fact. The metric definition in this case is:

\[ \text{Sum (Reg_Sls_Dlr) \{ ~ +, < Store+} \]
Ending fact and ending lookup grouping

When grouping by ending fact table or ending lookup table, the procedure is the same as that used in the case of beginning fact and lookup. The only difference is that the values selected for display represent the last occurrence instead of the first one. These options are only used with nonaggregatable metrics or facts.

The metric definition for an ending fact grouping is:

\[ \text{Sum (Reg_Sls_Dlr) \{ ~ +, >|Store+|} \]

Applying the end lookup grouping option calls for the use of lookup-table information, so that the last value used for display is an attribute element, not a fact. The metric in this case is defined as:

\[ \text{Sum (Reg_Sls_Dlr) \{ ~ +, > Store+|} \]

Ignore grouping

As in the case of filtering, this grouping option instructs the engine to omit the information in the aggregation. Using Region as the attribute level and Ignore as the value for grouping, the metric is defined as:

\[ \text{Sum (Reg_Sls_Dlr) \{ ~ +, !Region+|} \]

Additional level capabilities

In addition to the filtering and grouping functions described for metric dimensionality, MicroStrategy also offers the following options as advanced settings for metric definition:

- allow other users to add data to a metric level, which which is used to emulate MicroStrategy 6.x behavior and therefore affects only those projects that have been upgraded from 6.x
- add attributes in the filter to the level (dimensionality), which determines whether the metric filter is applied to the metric calculation
Both of these capabilities are enabled by default for metric definition. To disable them, use the syntax in the following examples:

- \( \text{Sum (Reg\_Sls\_Dlr) \{ ~+, Region+-} \)  
  Disallows additions to metric-level data

- \( \text{Sum (Reg\_Sls\_Dlr) \{ ~+, Region+;/} \)  
  Disallows direct application of report filters to metric calculations

- \( \text{Sum (Reg\_Sls\_Dlr) \{ ~+, Region+-;/} \)  
  Disallows both capabilities

### Pass-through functions

Pass-through expressions, also called Apply Functions, provide access to functionality that is not standard in MicroStrategy products but can be obtained through the relational database. When you include a pass-through expression in an attribute, fact, or transformation expression, the SQL Engine recognizes it as custom SQL and treats it as a pass-through expression. The pass-through expression is then sent to the relational database as written. For more information, see Appendix C, *Pass-through Expressions*. 
Introduction

Data mining generally refers to examining a large amount of data to extract valuable information. The data mining process involves using the predictive models based on existing and historical data to project potential outcome for business activities and transactions. MicroStrategy Data Mining Services facilitates the development and deployment of these predictive models.

This chapter introduces MicroStrategy Data Mining Services, which includes these features:

- Using MicroStrategy to create multi-variable regression predictive models
- Support for importing third-party predictive models using the PMML industry standard
- A Predictive Model Viewer that visualizes the predictive model
- A set of sample predictive metrics and reports incorporated into Customer Analysis Module (CAM)
In addition, this chapter describes the process of how to create and use predictive models with MicroStrategy and provides a business case for illustration.

Data Mining Services

The ultimate goal of data mining is to find hidden predictive information from a large amount of data. The data mining process involves using existing information to gain new insights into business activities by applying predictive models, using analysis techniques such as regression, classification, clustering, and association. By extending MicroStrategy’s powerful analytical, query, and reporting capabilities, MicroStrategy Data Mining Services can help answer business questions that would otherwise be too difficult to solve, allowing you to realize more value from your data.

Data Mining Services can be widely used in different industries and business areas, ranging from forecasting future results and customer behavior to classifying customers and estimating risk. A good example is an important area in marketing called Campaign Management. The goal of Campaign Management is to reduce the costs of marketing campaigns while increasing the positive response.

First, you gather data about the customers targeted for past campaigns, including information such as their age, gender, income, education, household size, and whether they responded positively or negatively to the campaigns. Next, you develop a MicroStrategy report to generate a dataset, which is then analyzed to determine if positive responders shared any factors. Once these predictive factors are identified and a predictive model is developed, a MicroStrategy metric is created that embodies this predictive model. This metric forecasts who is likely to respond positively to similar, future campaigns.

By using this metric in a report, you only need to contact those customers on the narrowed-down list, thereby lowering your company’s direct marketing costs and, in the meantime, increasing the effectiveness of the campaign. For details of this example, please see the last section of this chapter.
Approaches for data mining with MicroStrategy

There are a number of ways to incorporate MicroStrategy into the data mining workflow and these alternatives are best classified by how the model is ultimately deployed. Or, in other words, where does the scoring take place? There are three common approaches for incorporating data mining within MicroStrategy:

1 **Scoring the database**: Records are scored in batch and saved as tables or columns.

2 **Database does the scoring**: Database scores records in response to queries.

3 **MicroStrategy does the scoring**: MicroStrategy scores records using metrics and reports.

Each approach has both positive and negative aspects, but the good news is that MicroStrategy supports all three. The next section looks at these approaches in detail.

**Approach #1: Scoring the database**

In this approach, records are scored and inserted into the database as either new tables or new columns in existing tables. Most often, a third-party scoring engine receives a dataset and scores the records. Then the scores are added to the database. Once they are part of the database, MicroStrategy attributes or metrics can reference those scores, just like any other data in the database. Historically, this approach has been the most common and it has the following pros and cons:

**Pros:**

- Since a scoring engine does the scoring, model complexity and performance is hidden within the scoring engine. Thus, the scoring process does not require any database resources and does not impact other concurrent database work.

- At run time, data is simply read from the database without having to calculate the score on the fly.
• Avoids scoring records on the fly, which can slow down analysis of millions of scores.

• MicroStrategy can take advantage of this approach by just creating metrics or attributes for the scored data.

**Cons:**

• Requires database space and database administrator support.

• New records, inserted after the batch scoring, are not scored.

• Updating model or scores requires more database and database administrator overhead.

• In many companies, adding or updating information in the enterprise data warehouse is not done easily or on a whim. The cross functional effort required to score the database limits the frequency of scoring and prevents the vast majority of users from trying new models or changing existing ones.

This approach is really no different than adding other entities to a MicroStrategy project. For more information, see the *MicroStrategy Installation and Configuration Guide*.

**Approach #2: Database does the scoring**

In this approach, data mining features of the database system are used to perform the scoring. Nearly all major databases have the ability to score data mining models. The most common approach is to persist the model in the database and then generate scores by using extensions to the SQL queries processed by the database to invoke the model. A key feature of this approach is that the model can be scored in a system that is different from the data mining tool that developed the model.
The model is usually persisted in the database as a Predictive Model Markup Language, or PMML, object (more information on PMML can be found in later sections), or, less frequently, in some form of executable code. This is possible since the sophisticated algorithms needed to create the model are not required to score them. Scoring simply involves mathematical calculations on a set of inputs to generate a result. The ability to represent the model and score it outside of the model creation tool is relatively new, but more and more companies are adopting this approach, which has the following pros and cons:

**Pros:**

- Scores can be done "on the fly" even if new records are added.
- Updating the model is easier than the Score the database option.
- Requires less database space than the Score the database option.
- MicroStrategy can take advantage of this approach using its SQL Engine.

**Cons:**

- Requires database administrator support.
- Requires application knowledge of the database’s data mining tool (usually not the database administrator).
- Additional cost of the database data mining tool.

MicroStrategy has documented how to implement this approach for the IBM DB2 Intelligent Miner product. Please contact MicroStrategy Technical Support to get this Tech Note document.
Approach #3: MicroStrategy does the scoring

The previous two approaches both require support from the database and database administrators to implement data mining models. The time required and the potential for errors make database scoring more difficult than expected. This is especially true when the data mining process crosses organizational boundaries. Since the consumers of the model, say operations or marketing, usually do not have the ability to manipulate the database directly, they have to depend on someone else in IT to do the work.

They have to make sure the right model is used against the right data and that the results are available in the time and format needed by the consumer, not when and how it is convenient for IT. The overhead of cross-department coordination and the pressure to get results from more models, more frequently can make the entire data mining process break down in some companies.

Fortunately, there is a third approach for data mining that uses enterprise data resources without significantly increasing the overhead. MicroStrategy Data Mining Services is one of the newest features in the MicroStrategy platform and allows sophisticated data mining techniques to be applied directly within the business intelligence environment. Just as the other approaches, it also has pros and cons:

Pros:

- **MicroStrategy stores the predictive model in its metadata as a "predictive" metric that can be used just like any other metric.**
- **Scores can be done "on the fly" even if new records are added.**
- **Predictive Model can be viewed in MicroStrategy Desktop.**
- **Predictive Model is easily updated using Desktop Designer.**
- **Does not require database space nor database administrator support.**
• MicroStrategy can take advantage of this approach by using our Analytical Engine.

**Cons:**

• Does not take advantage of the database data mining features.

• Predictor inputs need to be passed from database to Intelligence Server. For large datasets, databases typically handle data operations more efficiently than moving data to MicroStrategy and scoring there.

A key enabler of this process is MicroStrategy’s ability to import predictive models, using PMML. Therefore, it is necessary for you to have some basic understanding of what PMML is and how it is used (see the following subsection).

**PMML**

*PMML* is an XML standard to represent data mining models. Developed by the Data Mining Group, DMG, an independent consortium consisting of over two dozen companies including MicroStrategy, PMML thoroughly describes how to apply a predictive model. PMML allows the use of a number of different model types, including Regression, Neural Networks, Clustering, Trees and Association, as well as support for data transformation and descriptive statistics. PMML is generated by nearly all data mining applications and workbenches, including SAS®, SPSS®, Oracle®, IBM®, Microsoft®, Teradata®, KXEN®, Quadstone®, Salford®, and others. MicroStrategy can import PMML and also generate PMML for certain types of models.

A major advancement for the industry, PMML allows the sophisticated, and sometimes esoteric, work of statisticians and data analysts to be easily deployed to other environments. PMML “closes the loop” between data mining tools and the applications that use data mining models. Several data mining and database vendors have announced integrations based on PMML. MicroStrategy is the first business intelligence platform to support the standard, and by allowing predictive metrics accessible to all users in the enterprise, MicroStrategy makes data mining for the masses possible.
For more information on PMML, please check the website of Data Mining Group (DMG), http://www.dmg.org, and other related documentation.

The Data Mining Services Workflow

The process of creating a predictive model and incorporating it into the MicroStrategy business intelligence platform involves the following steps:

1. Create a dataset report to be used to develop the predictive model.

2. Develop a predictive model from the dataset, using MicroStrategy or a third-party application.

3. Add the predictive model to your MicroStrategy project as a predictive metric.

4. Deploy the predictive metric in MicroStrategy reports to predict new outcomes, a process called scoring.

The predictive metric can also be used in filters, custom groups or wherever other MicroStrategy metrics are used.

A high-level description of each step is provided in the following sections of this chapter. Step-by-step instructions on the process can also be found in the online help.
Predictive metrics and performance

When it comes to scoring complex predictive models like neural networks, people often have the impression that it takes a long time to perform these analytical calculations. Actually, scoring predictive metrics does not significantly increase the time required to run reports. The chart below shows the impact of two different types of neural network models, one with 8 neurons and one with 41. As you can see, compared to an equivalent report with no predictive metrics, the reports with predictive metrics only takes about 10% longer to run or less.

For more information, please contact MicroStrategy Technical Support for the Tech Note that describes the impact of predictive metrics on performance.
Creating a dataset report

The first step in creating a predictive model is to develop a dataset report in MicroStrategy. It is recommended that you use MicroStrategy as the data mining source for the following reasons:

- **Enterprise data warehouse**: MicroStrategy projects typically use the enterprise data warehouse, which often contains clean, high-quality data. Since getting a good dataset is typically 50-75% of the data mining effort, using MicroStrategy can greatly reduce that effort.

- **Analytical functions**: It is easy to create new metrics using MicroStrategy’s vast library of over 200 analytical functions. From OLAP functions such as Rank, Percentile and Moving Averages to advanced statistical functions, raw facts in the database are easily turned into simple, and sometimes sophisticated, predictive inputs.

- **Relational databases**: MicroStrategy is optimized for all the major relational database vendors, which means that dataset reports are optimized to take advantage of these databases capabilities.

- **Security model**: MicroStrategy’s robust security model ensures that users can only access the data they are permitted to access. The MicroStrategy business intelligence platform can address privacy-preserving issues as the predictive model is developed and when it is deployed.

- **Easy handling of reports**: The dataset report can be easily created, refreshed, and accessed, even if it is large or contains complex calculations. Users do not have to be database administrators nor do they have to hand-code queries to access the data they need.

- **Consistent use of variable definitions**: The exact definition of variables that was used to develop the model is used again when the predictive model is deployed. This process is performed automatically, which ensures the results are consistent with the way the model was developed.

- **Data Sampling**: The dataset can be sampled from a large number of records.
Data mining datasets

Data mining datasets have a very simple structure. The data usually focuses on a specific subject or attribute, for example, customers, transactions, or products; this information is used to develop a predictive model.

A dataset is like a table in a database and usually has the following features:

- **Row**: Each row represents a specific attribute, such as customer, transaction, or product.

- **First column**: The first column is a unique identifier for the specific autoboot, such as customer name or identification number, transaction number, or product SKU number.

- **Remaining columns**: Each remaining column of the dataset contains data that describes the item in that row, such as customer age or annual purchases, transaction location or amount, or product color or cost. These columns either have the potential to be inputs to the predictive model or predictive inputs (also called independent variables), or they represent an outcome worth predicting (also called dependent variables).
The following is an example of a part of a dataset report for customer information:

<table>
<thead>
<tr>
<th>Customer</th>
<th>Customer ID</th>
<th>Customer Name</th>
<th>Customer Address</th>
<th>Customer Phone</th>
<th>Customer Email</th>
<th>Customer Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1001</td>
<td>John Smith</td>
<td>123 Main St.</td>
<td>555-1234</td>
<td><a href="mailto:johnsmith@email.com">johnsmith@email.com</a></td>
<td>Active</td>
</tr>
<tr>
<td>2</td>
<td>1002</td>
<td>Jane Doe</td>
<td>456 Oak St.</td>
<td>555-5678</td>
<td><a href="mailto:janedoe@email.com">janedoe@email.com</a></td>
<td>Closed</td>
</tr>
</tbody>
</table>

Notice that each attribute, such as Age Range, has two attribute forms on the report—the ID and the description. Some data mining software works better using numbers, such as the ID, while the description is included for ease of use.

Once the dataset is ready, it can be used in a data mining analysis, usually in one of the following ways:

- **Adding a metric to a MicroStrategy report:** A training metric is added to the report that trains a predictive model based on this data. More information on this approach can be found in *Creating a predictive model* in this chapter.
• **Using MicroStrategy data mart feature:** The dataset is persisted in the database as a table using MicroStrategy’s Data Mart feature. Third-party data mining applications can easily access databases using ODBC and the SQL query language. This setup also promotes consistency between the dataset used to develop the predictive model, especially for the variable names and datatypes. It does require database accessibility and database storage space, which could be a challenge depending on the organization. For more information on data marts, see *Data Marting*.

• **Exporting reports from MicroStrategy:** The dataset is exported to a particular file format using MicroStrategy’s export capabilities. Again, third-party data mining applications can access many file formats, such as Microsoft Excel and text files (CSV and so on). Exporting files means that the data type of each variable needs to be determined on the fly by the data mining application and that interpretation may need correction by the user. On the other hand, this approach is usually easier for most people and does not require help with database administration. See MicroStrategy online help for more information on exporting reports from MicroStrategy.

**Guidelines for creating a dataset report**

Before creating the dataset report, you need to make sure the attributes and metrics to be used as predictive inputs in the dataset report can also be used as inputs to the predictive metric. Recall that the predictive metric is the metric created from the predictive model after it is imported into MicroStrategy. By defining predictive inputs properly when building the dataset, you are guaranteed that the predictive model developed from that data receives the same inputs regardless how that model is used in MicroStrategy.
The following guidelines are intended to help you create your dataset report:

- **Use a flat report template.**

In the flat report template, place attributes on rows only and metrics on columns only. Since the dataset is a table consisting of a single header row followed by rows of data, placing attributes in the columns usually creates multiple header rows on each column, which cannot be easily represented in a database table.

- **Use only metrics as predictive inputs.**

The dataset report can contain attributes, which can be useful when you review the dataset. However, attributes cannot be used as predictive inputs because inputs to the predictive metric can only be other metrics. Neither attributes nor attribute forms can be used as inputs.

- **Create a metric using the attribute that is to be used as a predictive input.**

An “Attribute-based Predictive Inputs” metric represents the attribute form to be included in the model. It allows you to use an attribute as a predictor. For more information, see *Attribute-based predictive input metrics.*

- **Match each metric’s level with the attribute used on the rows of the dataset report.**

The attributes on the rows of the dataset set the dimensionality of the dataset report. If a metric is used in the predictive model without any dimensionality (also known as the level), its results changes based on the attributes of the report using the predictive metric. Creating another type of “predictive input” metric, one that sets the metric level, can resolve this problem. For more information, see *Metric-based predictive input metrics.*
• **Set a condition on the metric that provides the proper grouping, when grouping a metric’s results by an attribute.**

This condition is essentially a filter that allows an attribute to qualify metrics. For example, you could display customer revenue by payment method. For more information, see *Conditional predictive input metrics*.

• **Set the following property for the report:**

In the Report Editor, select Report Data Options from the Data menu, then in the Report Data Options dialog box, select Attribute Join Type under Calculations, and unchecked “Use Default”. Proceed to set the attribute join type of the report to “Preserve lookup table elements joined to the final pass result table based on the template attributes with/without filter” so that any missing data does not cause rows to be deleted. You can choose whether the filter is used (with) or not used (without).

### Predictive input metrics

A predictive input metric encapsulates the definition of another attribute or metric that can be used in predictive metrics. It serves several purposes, including:

• allowing attributes to be used as inputs to predictive metrics (see *Attribute-based predictive input metrics*).

• ensuring that the dimensionality of a metric is fixed, regardless of the context in which it is used (see *Metric-based predictive input metrics*).

• allowing a metric to be filtered, regardless of the context in which it is used (see *Conditional predictive input metrics*).
Attribute-based predictive input metrics

An attribute-based predictive input metric represents an attribute form to be included in the dataset. Attributes cannot be used as predictors in the dataset because the predictive metric accepts only metrics as inputs, not attributes or attribute forms.

Data mining often analyzes non-numeric, demographic, and psychographic information about customers, looking for attributes that are strong predictors. For example, your MicroStrategy project contains a Customer attribute with related attributes for age, gender, and income. To use these attribute forms in a predictive model, you would create an attribute-based predictive input metric for them. Such a metric for age would look like the following:

![Metric Editor](image)
To create an attribute-based predictive input metric

1  Create a new metric using the required attribute.

2  Specify the desired attribute form using the “Attribute@Form” format.

3  Since simple metrics like this one must have an aggregation function, place the attribute form in an aggregation function. Max is good to use since it returns the attribute form unchanged.

4  For dimensionality, add the attribute as a metric level so that this metric always returns.

5  Enable outer joins to include all data (in the Metric Editor, select Tools and then Metric Join Type).

6  Create a metric column alias to ensure the column name matches the metric’s name (in the Metric Editor, select Advanced Settings and then Metric Column Options).

7  Save the metric, using the alias from step 5 as the metric name.

Metric-based predictive input metrics

The attribute used on the rows of the dataset report sets the dimensionality of the data by restricting the data to a particular dimension, or level, of the data model. For example, if the Customer attribute is placed on the rows and the Revenue metric on the columns, the data in the Revenue column is at the customer level. If the Revenue metric is used in the predictive model without any dimensionality, then the data it produces changes based on the template of the report using the predictive metric. If Year is placed on the rows of that report, the predictive metric calculates yearly revenue rather than revenue for customers. Passing yearly revenue to a predictive model based on customer revenue yields the wrong results.
This problem can be easily resolved by creating a *metric-based predictive input metric* that matches the metric definition for Revenue but has its dimensionality set at the Customer level. Such a metric would look like the following.

![Customer Revenue - Metric Editor](image)

This approach is better than adding dimensionality to the Revenue metric itself because the metric may be used in other situations where the dimensionality should not be set to Customer.

---

**To create a metric-based predictive input metric**

1. Open the Metric Editor for the metric that requires dimensionality.
2. Add the necessary attributes as metric levels.
Enable outer joins to include all data (select Tools and then Metric Join Type).

Create a metric column alias to ensure that the column name matches the metric’s name (select Tools, then Advanced Settings, and then Metric Column Options).

Clear any “Break By” parameters that may exist on the metric’s function (highlight the function, right-click it, and then select the “Break By” tab).

Save the metric with the alias name from step 4.

Conditional predictive input metrics

To group a metric’s results by an attribute, you need to create a conditional metric for each category. For example, you want to use customer revenue by payment method in your data mining analysis. If you place the Customer attribute on the rows of the report, the Revenue metric on the columns, and the Payment Method attribute on the columns, you get the following report as a result:

<table>
<thead>
<tr>
<th>Customer</th>
<th>Payment Method</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Visa</td>
<td>Amex</td>
</tr>
<tr>
<td>Maxwell Aaronson</td>
<td>$485</td>
<td>$460</td>
</tr>
<tr>
<td>Hugh Abarca</td>
<td>$142</td>
<td>$639</td>
</tr>
<tr>
<td>Hazel Abelson</td>
<td>$260</td>
<td>$50</td>
</tr>
<tr>
<td>Brooks Abern</td>
<td>$739</td>
<td>$29</td>
</tr>
<tr>
<td>Ross Abram</td>
<td>$332</td>
<td>$130</td>
</tr>
<tr>
<td>Wylie Abrams</td>
<td>$107</td>
<td>$25</td>
</tr>
<tr>
<td>Don Addison</td>
<td>$403</td>
<td>$121</td>
</tr>
<tr>
<td>Merrell Adess</td>
<td>$510</td>
<td>$769</td>
</tr>
<tr>
<td>Keith Adler</td>
<td>$84</td>
<td>$997</td>
</tr>
<tr>
<td>Daniel Aguilar</td>
<td>$1,116</td>
<td>$120</td>
</tr>
<tr>
<td>Deborah Aguine</td>
<td>$1,067</td>
<td>$287</td>
</tr>
<tr>
<td>Dean Ahern</td>
<td>$1,229</td>
<td>$1,473</td>
</tr>
<tr>
<td>Elton Al-Timiu</td>
<td>$487</td>
<td>$132</td>
</tr>
<tr>
<td>Dorthy Alcaraz</td>
<td>$64</td>
<td>$266</td>
</tr>
<tr>
<td>Sarah Aldo</td>
<td>$663</td>
<td>$343</td>
</tr>
<tr>
<td>Peggy Alein</td>
<td>$1,680</td>
<td>$959</td>
</tr>
<tr>
<td>Maxwell Alexander</td>
<td>$79</td>
<td>$713</td>
</tr>
</tbody>
</table>
However, this report presents problems if it is used as a dataset report, because multiple headers are generated for all the columns (Revenue and each Payment Method). Besides, each column is really revenue for a particular payment method and, unless there is a metric that matches this definition, it is difficult to successfully deploy any model that uses one of these columns.

To solve this problem, you can create a **conditional predictive input metric** that filters Revenue for each Payment Method. This is essentially the same definition of the original Revenue metric, but has its conditionality set to filter revenue by a particular payment type.
To create a conditional predictive inputs metric

1. Create a filter for each of the necessary attribute elements.

2. For the example above, they are Payment Method = Visa, Payment Method = Amex, Payment Method = Check, and so on.

3. For each metric, create a metric-based Predictive Input metric (see above section).

4. Add a filter from Step 1 as a condition of the metric-based Predictive Input metric from Step 2 and save the metric.

5. Repeat for each filter created in Step 1.

The following report uses conditional predictive input metrics to generate the same results as the first report but in a dataset format.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Maxwell Aaronson</td>
<td></td>
<td>$485</td>
<td>$460</td>
<td>$13</td>
<td>$123</td>
<td>$274</td>
</tr>
<tr>
<td>Hugh Abarca</td>
<td></td>
<td>$142</td>
<td>$639</td>
<td>$979</td>
<td>$832</td>
<td>$163</td>
</tr>
<tr>
<td>Hazel Abelson</td>
<td></td>
<td>$260</td>
<td>$50</td>
<td>$87</td>
<td>$92</td>
<td>$163</td>
</tr>
<tr>
<td>Brooks Abern</td>
<td></td>
<td>$739</td>
<td>$29</td>
<td>$104</td>
<td>$63</td>
<td>$993</td>
</tr>
<tr>
<td>Ross Abram</td>
<td></td>
<td>$332</td>
<td>$130</td>
<td>$18</td>
<td>$142</td>
<td>$511</td>
</tr>
<tr>
<td>Wylie Abrams</td>
<td></td>
<td>$107</td>
<td>$25</td>
<td>$28</td>
<td>$524</td>
<td>$16</td>
</tr>
<tr>
<td>Don Addison</td>
<td></td>
<td>$403</td>
<td>$121</td>
<td>$1,129</td>
<td>$459</td>
<td>$117</td>
</tr>
<tr>
<td>Merrell Adess</td>
<td></td>
<td>$510</td>
<td>$769</td>
<td>$153</td>
<td>$141</td>
<td>$107</td>
</tr>
<tr>
<td>Keith Adler</td>
<td></td>
<td>$84</td>
<td>$997</td>
<td>$116</td>
<td>$169</td>
<td>$806</td>
</tr>
<tr>
<td>Daniel Aquilar</td>
<td></td>
<td>$1,116</td>
<td>$120</td>
<td>$77</td>
<td>$329</td>
<td>$1,724</td>
</tr>
<tr>
<td>Deborah Aquire</td>
<td></td>
<td>$1,067</td>
<td>$287</td>
<td>$146</td>
<td>$235</td>
<td>$413</td>
</tr>
<tr>
<td>Dean Ahern</td>
<td></td>
<td>$1,229</td>
<td>$1,473</td>
<td>$36</td>
<td>$165</td>
<td>$244</td>
</tr>
<tr>
<td>Elton Al-Timinu</td>
<td></td>
<td>$487</td>
<td>$132</td>
<td>$355</td>
<td>$63</td>
<td>$101</td>
</tr>
<tr>
<td>Dorthy Alcaraz</td>
<td></td>
<td>$64</td>
<td>$266</td>
<td>$129</td>
<td>$73</td>
<td>$27</td>
</tr>
<tr>
<td>Sarah Aldo</td>
<td></td>
<td>$663</td>
<td>$343</td>
<td>$51</td>
<td>$406</td>
<td>$72</td>
</tr>
<tr>
<td>Peggy Alein</td>
<td></td>
<td>$1,680</td>
<td>$959</td>
<td>$180</td>
<td>$1,151</td>
<td>$812</td>
</tr>
<tr>
<td>Maxwell Alexander</td>
<td></td>
<td>$79</td>
<td>$713</td>
<td>$112</td>
<td>$880</td>
<td>$176</td>
</tr>
</tbody>
</table>
Using non-MicroStrategy datasets

While there are benefits to starting with a dataset from MicroStrategy, there are cases where the dataset used to create predictive models came from other sources. This could happen for a number of reasons. Perhaps the data mining process started with data pulled from multiple, different systems. Perhaps the data was transformed or processed in some way by the data mining tool. Or, perhaps the data mining process was used to determine which data should be added to the database.

In any case, models that were developed from other data can still be used with MicroStrategy, so long as a couple of requirements are met:

1. The data that forms the inputs to the data mining model must be present in MicroStrategy as predictive input metrics.

2. The “meaning” of the data is unchanged when used in MicroStrategy, including any dimensionality requirements. For example, if the data mining model expects “Recent Customer Transaction Count” as input, the definitions for recent, customer, transaction, and count must be the same in MicroStrategy, and yield the same results under the same conditions used when developing the model.

As you can see in Importing the predictive model, if the model expects an input that is not already defined in MicroStrategy, the user is warned and prompted to select a metric that matches. If one does not exist, the import process is aborted and that metric needs to be added to the project before the model can be imported.
Creating a predictive model

After you have created a dataset report, the next step is to create a predictive model, which will then be imported into MicroStrategy. You can create a predictive model in one of the following two ways:

- Using a third-party data mining application
- Using a built-in MicroStrategy function

Using third-party applications

There are many companies that offer data mining applications and workbenches. These tools are specialized for the data mining workflow and are usually rich in data mining features and options. MicroStrategy Data Mining Services integrates seamlessly with these applications in the following ways:

1. MicroStrategy is used as the source of data mining dataset.

2. After the third-party application develops the PMML-based predictive model based on the dataset, MicroStrategy imports the model to generate reports.

You can use one of the following two options to provide the MicroStrategy dataset to third-party applications:

- Save the dataset as a data mart using MicroStrategy’s data mart feature. For more information on data marts, see Data Marting.

- Export to a particular file format using MicroStrategy’s export capabilities. See the MicroStrategy online help for more information on exporting reports from MicroStrategy.
Creating a predictive model © 2005 MicroStrategy, Inc.

Using MicroStrategy

While there are many sophisticated data mining algorithms, some data mining techniques are fairly simple and do not require specialized tools. The MicroStrategy TrainRegression function can be used to create predictive models using the regression data mining technique. This technique should be familiar to you if you have ever tried to extrapolate or interpolate data, or tried to find the line that best fits a series of data points, or used Microsoft Excel’s LINEST or LOGEST functions.

Regression analyzes the relationship between several predictive inputs, or independent variables, and a dependent variable that is to be predicted. It does this by finding the line that best fits the data, with a minimum of error.

For example, you have a dataset with just two variables, X and Y, which are plotted as in the following chart:
Using the Regression technique, it is relatively simple to find the straight line that best fits this data (see the following chart). The line is represented by a linear equation in the classic $y = mx + b$ format, were $m$ is the slope and $b$ is the $y$-intercept.
Alternatively, you can also fit an exponential line through this data (see the following chart). This line has an equation in the $y = b m^x$ format.

So, how can you tell which line has the better fit? There are many statistics used in the Regression technique. One basic statistic is an indicator of the “goodness-of-fit,” meaning how well the line fits the relationship among the variables. It is also called the Coefficient of Determination, whose symbol is $R^2$. The higher the $R$, the better the fit. We can see that the linear predictor has $R^2 = 0.7177$ and the exponential predictor has $R^2 = 0.7459$; therefore, the exponential predictor statistically is a better fit.

With just one independent variable, this example is considered a “univariable regression” model. In reality, the Regression technique can work with any number of independent variables, however, always only one dependent variable. While the “multivariable regression” models are not as easy to visualize as the univariable model, the technique will generate statistics so you can determine the goodness-of-fit.
The MicroStrategy TrainRegression function allows both types of Regression techniques. Both the linear and exponential Regression predictor can be created easily, with just one change to set the type of model.

**Example: Predicting quarterly online sales**

First, let us create a metric using the TrainRegression function. As you can see from the following, the Train Regression metric has a set of metrics as its inputs, including the On-line Sales metric we are trying to predict (the dependent variable) and several metrics that describe each quarter (the independent variables). The order of these inputs in the metric’s expression does not matter.
Next, set the parameters for the TrainRegression metric for it to work properly. Highlight the TrainRegression portion of the expression, right-click it and view the TrainRegression parameters, shown as follows.

The TrainRegression Parameters have the following meanings:

- **MinimumR2**: This value is used to determine which independent variables will be included in the final regression equation. The $R^2$ value is a number between 0 and 1. A value of 0 indicates no predictive value, whereas a value of 1 indicates high predictive value. The $R^2$ value is calculated for each independent variable, in relation to the dependent variable. If the $R^2$ value is less than the specified Minimum $R^2$, or 0, it will not be included in the generated regression equation. If all the predictive inputs fail to calculate a high enough $R^2$ to be included, no model will be generated and you’ll see a “-1” for each row and the training metric’s column in the training report.
• **ModelFileName:** This is the physical or relative file name to be created by TrainRegression. If set to, for example, \temp\model.xml, the output PMML will be written to that file. It is best to always use the .xml suffix. Also, on systems where the MicroStrategy Intelligent Server is remote, a shared drive reference (\System1\model.xml) will ensure that the PMML file will be accessible to MicroStrategy Desktop, which is necessary for importing the model.

• **ModelName:** This name will be incorporated into the output PMML.

• **ModelPropName:** This parameter can be useful if you want to perform this process programmatically. If you were to write a VB program that set up and executed a training report, this parameter allows you to assign a name to the output PMML property, which is part of the report. After the report has been executed, the program can read the output PMML from this property.

• **RegressionType:** TrainRegression supports two types of regression:

  **Multiple Linear Regression (MLR):** If this type of regression is specified, the function will attempt to calculate the coefficients of a straight line that best fits the input data. The calculated formula follows the following format: $y = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$, where $y$ is the target value, and $x_1 \ldots x_n$ are the independent variables. The MLR technique finds the $b_n$ values that best fit the data. To use MLR, set RegressionType = 0.

  **Multiple Exponential Regression (MER):** If this type of regression is specified, the function will attempt to calculate the coefficients of an exponential curve that best fits the input data. This is accomplished by calculating the natural log (ln) of the input target variables and then performing the same calculations used for MLR. Once the straight-line coefficients are calculated, MER takes the natural exponential of the values, which results in the coefficients for the formula
of an exponential curve. The calculated formula will follow the following format: \( y = b_0 \cdot (b_1 \cdot x_1) \cdot (b_2 \cdot x_2) \cdot \ldots \cdot (b_n \cdot x_n) \), where \( y \) is the target value, and \( x_1 \ldots x_n \) are the independent variables. The MER technique finds the \( b_n \) values that best fit the data. To use MER, set RegressionType = 1.

- **Target:** This is the name of the dependent variable. This name must correspond to one of the arguments of the TrainRegression function. The case-sensitive name will be incorporated into the output model.

- **Variables:** This is a list composed of the names of all variables (independents and dependent) in the same order as they are passed into the TrainRegression function. The names should be separated by commas, with no leading or trailing blanks. These case-sensitive names will be incorporated into the output PMML.

Once you have created a metric with one regression type, it is easy to make another metric of the other type by simply copying the metric and changing the RegressionType parameter. It would be good to also change the ModelFileName and ModelName parameters so there is not any conflict with the original training metric.

To use the training metric, add it to your dataset report and run the report. When the report execution finishes, the training metric’s column will contain the results of the Regression model, which is saved to the location specified by the ModelFileName parameter.

---

**To create a predictive model using MicroStrategy**

1. Log on to a MicroStrategy project.

2. Open the Metric Editor

3. Using the object browser, locate the TrainRegression function, and add it to the metric formula.

4. Add each independent variable to the TrainRegression argument list.
5. Add the dependent variable to the TrainRegression argument list (order-independent).

6. Highlight and right-click the TrainRegression function name.

7. Select TrainRegression parameters from the pop-up menu.

8. Fill in the appropriate values for each of the function parameters.

9. Click **Save and Close** on the toolbar. Select a location for the new metric and enter a metric name.

10. Close the Metric Editor.

11. Add the training metric to a training report.

12. Execute the report.

It is recommended that reports containing training metrics have report caching disabled. The reason for this is to ensure that the PMML model is always generated. If the report is in the report cache, there is no need for MicroStrategy to execute the training metric, since the results have already been cached. Therefore, the PMML model file is not generated. To disable report caching for a training report, open the report with the Report Editor and select **Report Caching Options** from the Data menu. This opens the Report Caching Options dialog box, which allows you to disable report caching for the report.

When the report has completed execution, a file containing the generated PMML model can be found at the location specified by the training metric. This model can now be imported into a MicroStrategy project, which is the subject of the next section.
Importing the predictive model

After you have created a predictive model and generated PMML to represent that model, the next step is to import it into your MicroStrategy project. Instructions are provided in the following as well as in the online help.

Before you can access the Import Data Mining Model option, you must have the “Create Application Objects” and “Use Metric Editor” privileges. For information on viewing or changing your privileges, see the online help.
To import a predictive model into MicroStrategy

1. On Desktop, select **Import Data Mining Model** from the Tools menu.

2. Select the PMML file to import by clicking the “…” button.

3. Name the model.

4. Enter a description of the model.

5. Select the aggregation function. For more details, see “Aggregating predictive metrics” starting on page 349.

6. Click **OK** to import the model.

7. Select a folder where the predictive metric is saved.

The import feature will read the PMML automatically and create the predictive metric in the folder specified. During the importing process, several key model characteristics are determined from the PMML, including the following:

- **The type of data mining algorithm**: Based on this, the appropriate MicroStrategy data mining function is used in the expression of the metric.

- **The inputs to the data mining function**: This is where setting the column alias of the prediction input metric to match its name really pays off. Since Data Mining applications use this name to identify each variable, we can use the name of each variable in the PMML to identify which MicroStrategy metric should be used. The import feature robustly handles all potential circumstances in this area as follows:
  
  - There is only one MicroStrategy metric with that name: That metric will be used as an input to the function automatically.
  
  - There is more than one MicroStrategy metric with that name: The user will be prompted to select the right metric.
There is no MicroStrategy metric with that name: The user will be prompted to select a metric from within the project, or the user can cancel the import and try again later.

- **The version of PMML**: Since different versions of PMML have different specifications, the version is identified so models are processed properly.

- **XML schema validation**: The PMML is validated against its XML schema definition to ensure it conforms to the standard. This greatly reduces the possibility of problems when the model is processed. If the PMML does not validate, the user will be informed of the error, and the import process will not complete.

- **Model verification**: If the PMML contains model verification data, that data is used to determine if MicroStrategy generates the expected results for the model. If not, the user is informed of results and has the option to continue or to cancel the import process.

Also, the predictive metric has a unique icon so it is easy to identify and distinguish it from the other metrics.
Data mining function parameters

Each data mining function in MicroStrategy has a set of parameters (shown as follows) that describe the model or tell how to return results:

- **Confidence**: This value is used to determine whether the predictive metric returns a score or a confidence. If it is set to False, then it returns a Score. If it is set to True, it returns a confidence. If the predictive metric has more than two outcomes, the Target parameter is used to determine which confidence to return (see **Target** below for more information).

- **ModelID**: This is an internal value used to uniquely identify each model. Since more than one metric can contain the same model, MicroStrategy Intelligence Server uses this ModelID to identify identical models. This value should not be changed under normal circumstances.

- **PMML**: This is the actual PMML that describes the predictive model. This value should not be changed under normal circumstances.
• **PMMLVersion**: This is the version of PMML used by the predictive model. This value should not be changed under normal circumstances.

• **Target**: When the Confidence parameter is set to True and the predictive metric has more than two outcomes, this field can be used to specify which confidence to return by setting its value to the desired outcome, or leaving it blank to return the confidence in the outcome that is scored.

### Returning confidences/probabilities instead of scores

When a PMML model is imported, the resulting metric is configured to calculate the prediction, or score, specified by the model. For example, if the model predicts whether a customer is likely to respond to a marketing campaign, the predictive metric that is created will, by default, return a score that says whether the customer is expected or not expected to respond (for example, Yes or No, 0 or 1, and so on). It is often useful to return the probability or confidence in that score, rather than the score itself.

This can be easily done by making a copy of the predictive metric and changing some parameters on its function. Each data mining function has a Confidence parameter that is set to False as default. To return a confidence instead of a score, change the Confidence value to True. If the model's output is binary (for example, Yes or No, 0 or 1, and so on), that is all that is needed to return a confidence in the outcome. For models with more than two possible outcomes (for example, Low, Medium, High), there is a Target parameter. With Confidence set to true, the Target parameter determines which confidence to return. If the Target parameter is blank, it will return the confidence in the resulting score. To get the confidence in a particular outcome, set the Target parameter to that outcome. For example, to return the confidence in the High outcome, set the Confidence parameter to True and set the Target parameter to High.

When the Confidence parameter is false, the Target parameter is ignored.
Also note that for some types of models, especially some variations of regression models, there is no way to provide a confidence number. In these cases, MicroStrategy will display "-1" as an indicator that a confidence cannot be calculated.

Aggregating predictive metrics

One of the most powerful features of integrating data mining models with a business intelligence platform is the ability to drill up and down through layers of data. In order for predictive results to make sense, we need to specify the aggregation function to be used.

There are actually two steps that need to be taken to ensure the predictive metric aggregates properly under all circumstances:

- Placing the predictive function within an aggregation function
- Specifying the dynamic aggregation function on the predictive metric (required only when the predictive metric is used with MicroStrategy OLAP Services)

Choosing proper aggregation function requires some knowledge about how the model behaves. For example,

- if the predictive metric generates a score that is a zero or a one, use Sum to calculate the number of "one" scores.
- if the predictive metric generates a "linear" output, like "Forecasted Revenue," usually from a regression predictor, use Sum to roll up the predictive results.
- if the predictive metric generates a confidence or percentage, use Average to calculate the mean confidence.
- if the predictive metric generates a numeric classifier, like a cluster/segment number, use Mode to calculate the most common classifier.
- for models with outputs that cannot be aggregated, select "None" as the aggregation function.
While using an explicit aggregation function is useful for normal reports, deployments that take advantage of MicroStrategy OLAP Services should also set the predictive metric's dynamic aggregation function. Desktop's Import Data Mining Model feature will do this automatically but you can set the dynamic aggregation function using the metric editor.

**To set the dynamic aggregation**

1. Edit the predictive metric using the Metric Editor.
2. Select the Subtotals/Aggregation tab.
3. Change the Dynamic Aggregation function as appropriate.

**Using the predictive metric**

**Using the predictive metric in reports**

With the predictive model implemented as a metric, it can be used in reports to determine possible trends and outcomes. Again, creating a report for data mining is similar to creating a regular report. These reports have no special requirements, other than including the predictive metric. For more detailed instructions on creating a report, see the MicroStrategy Online Help.

Predictive models are generated based on a certain level of input data. Therefore, in order for a model to produce valid results when scored, the model must be provided the same level of input data. If the user of the model does not have access to this level of data (for example, a security filter on the user does not allow such access) then the model cannot be expected to return valid scores.
Using the predictive metric in other objects

In addition to reports, predictive metrics can also be used in other objects. Generally, wherever a regular MicroStrategy metric is used, a predictive metric can be used. The objects where predictive metrics can be used include:

- Filters
- Custom Groups
- Consolidations
- Derived Metrics
- Shortcut Metrics

Predictive Model Viewer

Predictive metrics are different from other MicroStrategy metrics since they implement a data mining model, which is more than a mathematical expression. These predictive models, and the PMML that describes them, contain information about variables, variable transformations, and details about the data mining techniques they implement. All this information can be viewed on MicroStrategy Desktop using the Predictive Model Viewer.

You can access the Predictive Metric Viewer in one of the following ways:

- On Desktop, right-click a predictive metric in the right-hand pane and select View Predictive Model.
- In the Metric Editor, when editing a predictive metric, select View Predictive Model from the Tools menu.
- In a Desktop report or graph, right-click a predictive metric in the right-hand pane and select View Predictive Model. For more information on how to use this viewer, please see the MicroStrategy online help.
Example

Recall the campaign management scenario described at the beginning of this chapter. Your company wants to improve the effectiveness of its marketing campaigns, with the goals of reducing costs and increasing the percent of positive responses. The results of a previous campaign will be analyzed to determine what factors, if any, can be used to predict the performance of a similar future campaign.

The previous campaign was run during the fall of 2002, when 78 customers out of 359 responded favorably. Accurate data on the targeted customers and their responses have been entered into a MicroStrategy project.

The first step in data mining is to decorate a dataset report. The pertinent information for this report is listed as follows:

- Name
- Age
- Education
- Gender
- Household count
- Income range
- Marital status

The dataset report for this example can be found in the Creating a dataset section.

You want to use all of these attributes, except for customer name, as predictors in the predictive model. Therefore, you must create a "Predictive Inputs" metric for each, since the predictive metric accepts only metrics as inputs. Some example "Predictive Inputs" metrics for this report are as follows:

\[
\begin{align*}
\text{Max}([\text{Customer Age Range}]@ID) & \{\text{Customer}\} \\
\text{Max}([\text{Customer Age Range}]@DESC) & \{\text{Customer}\} \\
\text{Max}([\text{Customer Education}]@DESC) & \{\text{Customer}\}
\end{align*}
\]
Filter the dataset report to include only the customers acquired before October 1, 2002, since newer customers would not have been included in the fall 2002 campaign.

Once the dataset report is complete, use it to generate the dataset.

After analyzing the dataset with a third-party data mining application, a predictive model is developed and the PMML representation of that model is imported into MicroStrategy. Even though a large number of attributes were included, the resulting model includes only those attributes that were established as strong predictors—gender, age, education, and household count. The predictive metric is shown as follows:

The predictive metric actually uses a neural network algorithm to score each record and determine if that customer is likely to respond or not. Using this predictive metric, we can validate it against the original data in a report like the one below, which compares the actual response with the response calculated by the predictive metric.
This report shows the actual (Responded to Campaign) and predicted (Response Predictor) results for the 359 customers who were targeted in the first campaign. It also shows each customer's Response Propensity, which can be thought of as the probability or confidence that a customer will respond favorably to the campaign. Finally, the Correct? column indicates if the Response Predictor is correct or not.

As you can see from the data, the Response Predictor metric correctly predicted 344 out of 359 responses, which corresponds to about 95 percent accuracy. This accuracy is definitely acceptable for marketing purposes. Note that one of the incorrectly predicted customers, Marion Kelner, actually did respond but the Response Predictor says she would not.
That score simply uses a Response Propensity threshold of 50% to determine whether a customer is likely to respond (greater than 50%) or not (less than 50%). We can see that with a Response Propensity of 45%, we could have correctly predicted Marion Kelner if the threshold was set lower.

<table>
<thead>
<tr>
<th>Customer</th>
<th>Responded</th>
<th>Response</th>
<th>Response</th>
<th>Correct?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>78</td>
<td>359</td>
<td>359</td>
<td>344</td>
</tr>
<tr>
<td>Count</td>
<td>359</td>
<td>359</td>
<td>359</td>
<td></td>
</tr>
<tr>
<td>Carole Vomdran</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Cleatus Kristick</td>
<td>No</td>
<td>No</td>
<td>3%</td>
<td>OK</td>
</tr>
<tr>
<td>Dominick Onell</td>
<td>Yes</td>
<td>Yes</td>
<td>100%</td>
<td>OK</td>
</tr>
<tr>
<td>Hector Pelino</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Melanie Searles</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Monica Neptune</td>
<td>Yes</td>
<td>Yes</td>
<td>100%</td>
<td>OK</td>
</tr>
<tr>
<td>Mai Normandin</td>
<td>No</td>
<td>No</td>
<td>1%</td>
<td>OK</td>
</tr>
<tr>
<td>Iris Vaughn</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Enzo Cutting</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Detyne Ferber</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Conrad Hughes</td>
<td>No</td>
<td>No</td>
<td>1%</td>
<td>OK</td>
</tr>
<tr>
<td>Jan Taylor</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Iris Zeller</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Carole Robison</td>
<td>Yes</td>
<td>Yes</td>
<td>100%</td>
<td>OK</td>
</tr>
<tr>
<td>Evangeline Marrocco</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Marion Kelner</td>
<td>Yes</td>
<td>No</td>
<td>45%</td>
<td>Error!</td>
</tr>
<tr>
<td>Chat Saxton</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Emanuelle Schneider</td>
<td>Yes</td>
<td>Yes</td>
<td>100%</td>
<td>OK</td>
</tr>
<tr>
<td>Jordan Patterson</td>
<td>No</td>
<td>No</td>
<td>4%</td>
<td>OK</td>
</tr>
<tr>
<td>Arley Corbett</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Susan Vansickle</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Melodi Kelner</td>
<td>Yes</td>
<td>Yes</td>
<td>100%</td>
<td>OK</td>
</tr>
<tr>
<td>Chat Goldberg</td>
<td>No</td>
<td>No</td>
<td>0%</td>
<td>OK</td>
</tr>
<tr>
<td>Erling Searles</td>
<td>No</td>
<td>No</td>
<td>3%</td>
<td>OK</td>
</tr>
<tr>
<td>Fannie Mappala</td>
<td>Yes</td>
<td>Yes</td>
<td>100%</td>
<td>OK</td>
</tr>
<tr>
<td>Frederic Butcher</td>
<td>No</td>
<td>No</td>
<td>43%</td>
<td>OK</td>
</tr>
</tbody>
</table>
Another way to look at this model is by using a Lift Chart. Lift Charts show the gain, or lift, possible when using the predictive metric to target customers over selecting customers at random.

This chart plots the percent of customers available to be part of the campaign against the percent of customers that responded to the campaign. The red dashed line shows that selecting a percent of customers at random will result in an equivalent percent of responders. For example, randomly contacting 20% of the customers will net 20% of responders, all things being equal.

On the other hand, the solid red line shows the benefit of using the Response Predictor metric. Using this predictor, the most likely responders can be targeted first, providing a "lift" to the campaign's results. In other words, using the Response Predictor to identify the customers most likely to respond favorably, contacting 20% of the customers can yield over 80% of the responders, a gain of over four times the random approach! Marketing teams use lift charts like this to define programs that optimize the costs and the returns of their campaigns. Finally, you can use the metric to predict the responses of the customers who were not used in developing the model, that is, the customers acquired on or after 10/1/2002.
The following report shows that out of 1,081 new customers, 221 are likely to respond positively to a campaign similar to that of fall 2002. Based on these results, the number of customers targeted can be greatly reduced from 1,083 to 221. Costs for the campaign will decrease while positive responses will likely increase significantly.
CUSTOM GROUPS AND CONSOLIDATIONS

Introduction

A custom group is a set of special filters that can be placed on a template. It is made up of an ordered collection of elements called custom group elements. Consolidations are used to specify the data you want to view in your report. They allow you to group attribute elements in new ways without changing the metadata and warehouse definitions. Both of these features allow you to qualify a report on a row-by-row basis.

This chapter describes the significance of custom groups and consolidations. It will build on the topics of filters, attributes, and attribute elements.
Custom groups

A custom group is an object that can be placed on a template and is made up of a collection of elements called custom group elements. Each element contains its own set of filtering or banding qualifications.

Each custom group element can be labeled with a meaningful header and can include a logical expression containing any of the following:

- attribute qualification
- set qualification
- report object
- filter object
- custom group banding qualifications

A custom group element can also use combinations of any of the components listed above, except custom group banding qualifications.

This set of qualifications resolves into a list of attribute elements after the report is run. Custom groups, therefore, provide a method to group attribute elements from the same or different attributes to meet your reporting requirements.

For example, using the Custom Group Editor, you can create the custom group Store Inventory as follows:

**Store Inventory**

Small stores with low inventory

\[
\text{Store Sales < 50} \\
\text{AND} \\
\text{Store Inventory < 200}
\]

Large stores with low inventory

\[
\text{Store Sales > 50} \\
\text{AND} \\
\text{Store Inventory < 200}
\]
Depending on the options you select in the Custom Group Editor, the custom group could appear on the report as shown below.

![Custom group element headers](image)

<table>
<thead>
<tr>
<th>Small stores with low inventory</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>store &quot;a&quot;</td>
<td>22</td>
</tr>
<tr>
<td>store &quot;t&quot;</td>
<td>16</td>
</tr>
<tr>
<td>Large stores with low inventory</td>
<td>258</td>
</tr>
<tr>
<td>store &quot;x&quot;</td>
<td>98</td>
</tr>
<tr>
<td>store &quot;m&quot;</td>
<td>160</td>
</tr>
</tbody>
</table>

The output level of a custom group (in this example, Store) is based on the filtering conditions of the element. Each element in the custom group can have a different output level. For example, elements are completely independent. The fact that they are included in the same custom group means only that they are displayed on the same report.

**Benefits of using a custom group**

The benefit of a custom group is its ability to group attribute elements in a way that is not defined in the data warehouse. You can create “relationships” between the attribute and the custom group. A custom group can organize attribute elements through:

- attribute qualification
- set qualification
- report
- filter
- banding
• advanced qualification

Refer to Chapter 5, *Filters*, for more information on these qualification types.

Currently, a report limit defined on a custom group report is ignored.

**Banding qualification**

Banding qualifiers enable you to create banding custom group elements. *Banding* is a method of slicing a list, defined by the output level, of attribute elements using the values of a metric. For example, you can slice the list of stores (“Store” attribute elements) using the values of the metric “Total Sales.” Suppose you have created a report that ranks stores according to the revenue generated by each store. You might decide to group the stores by creating one group for the top 10 stores, a second group for stores 11-20, and a third group for stores 21-30.

You can apply different types of banding:

• Band size: to slice the range of metric values defined by “start at” and “stop at” values into a number of bands, each defined by the parameter “step size.”

For example, in the following diagram the “start at” value is 10, “stop at” is 50, and “step size” is 10. These settings slice the group into four bands.
Band count: to define the number of equal bands into which the range of metric values is sliced. On the other hand, you use band size to define the size of each band.

For example, in the preceding diagram, the band count is set to four, which has the same effect as the band size example. If you set the band count to five instead, eight bands are produced.

Banding points: to specify the value where a band is placed. This enables you to produce bands of different sizes.

The engine uses its internal logic to create the bands based on the banding points that you specify.

For example, you want to create a report with two bands, one band showing the top 10 stores and the second band showing stores 11-100. For this, you must use three points—1, 10, and 100—as shown in the following figure.

To show a band of the bottom 10 stores and then the remaining, use the same points in reverse order, that is, 100, 10, and 1.
Example: banding points

Report requirements

You want to create two reports that rank employees according to the revenue that each employee generates.

1. Based on their revenue, the first report should segregate the employees into the following three groups; Top for the top 10%, Next for the next 40%, and Lowest for the lowest 50%.

2. Based on their revenue, the second report should segregate the employees into the following three groups; Lowest for the lowest 10%, Next for the next 40%, and Top for the top 50%.

Solution

To create the first report, create a custom group called Employee Revenue and specify the banding points as 0, 10, 50, and 100. Create a report that uses this custom group. A sample report is shown in the following figure.
The engine considers 0 as the highest banding point and 100 as the lowest banding point; hence, based on the revenue, it places the highest 10% of employees in the first band and so on.

To create the second report, create a custom group called Employee Revenue and specify the banding points in reverse order, that is, specify the banding points as 100, 90, 50, and 0.
Again, the engine applies the same logic. It considers 0 as the highest banding point and 100 as the lowest banding point. Based on the revenue, it places the lowest 10% of employees in the first band, the next 40% of employees in the second band, and the highest 50% of employees in the third band as shown in the following figure.

<table>
<thead>
<tr>
<th>Employee Revenue</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td></td>
<td>$16,936,434</td>
</tr>
<tr>
<td>Lowest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kelly Laura</td>
<td></td>
<td>$1,285,585</td>
</tr>
<tr>
<td>Savvyer Leanne</td>
<td></td>
<td>$316,786</td>
</tr>
<tr>
<td>Tarrison Mary</td>
<td></td>
<td>$333,377</td>
</tr>
<tr>
<td>Young Sarah</td>
<td></td>
<td>$305,534</td>
</tr>
<tr>
<td>Next</td>
<td></td>
<td>$5,121,849</td>
</tr>
<tr>
<td>Bernstein Lawrence</td>
<td></td>
<td>$403,122</td>
</tr>
<tr>
<td>Conner Beatrice</td>
<td></td>
<td>$380,503</td>
</tr>
<tr>
<td>Carcoran Peter</td>
<td></td>
<td>$346,565</td>
</tr>
<tr>
<td>Folks Adrienne</td>
<td></td>
<td>$418,967</td>
</tr>
<tr>
<td>Hall David</td>
<td></td>
<td>$358,542</td>
</tr>
<tr>
<td>Hunt Matthew</td>
<td></td>
<td>$418,415</td>
</tr>
<tr>
<td>Ingles Walter</td>
<td></td>
<td>$377,583</td>
</tr>
<tr>
<td>Keiferson Jack</td>
<td></td>
<td>$399,860</td>
</tr>
<tr>
<td>Lynch Sam</td>
<td></td>
<td>$391,809</td>
</tr>
<tr>
<td>Smith Thomas</td>
<td></td>
<td>$420,931</td>
</tr>
<tr>
<td>Sonder Melanie</td>
<td></td>
<td>$421,036</td>
</tr>
<tr>
<td>Yager Beth</td>
<td></td>
<td>$362,742</td>
</tr>
<tr>
<td>Zemlicka George</td>
<td></td>
<td>$431,746</td>
</tr>
<tr>
<td>Top</td>
<td></td>
<td>$10,529,000</td>
</tr>
</tbody>
</table>

Example: banding points
Custom group elements

A custom group element is a logical expression of qualifications. A custom group element contains:

- A name or header—This is an arbitrary name you define when you create the element. This name can be displayed on the report and can be modified as desired. The Custom Group Editor provides you with different options for displaying the header. Because the custom group element can appear on the report, choose a significant name for the grouping of elements that you are defining.

- An expression or qualification—You can define any qualification, logical expression, or banding of the qualification, or you can use previously created filters to build the custom group element.

For example, you can create the custom group Store Inventory as follows:

**Store Inventory**

Small stores with low inventory

    Store Sales < 50  
    AND
    Store Inventory < 200

Large stores with low inventory

    Store Sales > 50  
    AND
    Store Inventory < 200

The custom group elements in this example are:

Small stores with low inventory, which is a logical expression of the following two metric qualifications (MQs):

    Store Sales < 50  (MQ1)  
    AND
    Store Inventory < 200  (MQ2)
and Large stores with low inventory, which is a logical expression of the following two metric qualifications:

\[
\begin{align*}
\text{Stores Sales} & \gt 50 & (\text{MQ1}) \\
\text{AND} & \quad (\text{logical operator}) \\
\text{Store Inventory} & \lt 200 & (\text{MQ2})
\end{align*}
\]

### Custom group element headers

A custom group is composed of one or more custom group elements and custom group element headers. For each individual grouping of custom group elements, there is a corresponding header. The header is used as an identifier on the report row or column. The Custom Group Editor provides you with different options for displaying the header.

### Custom group options

Custom groups give a natural hierarchical structure to their elements. Each custom group element can be viewed as a set of smaller grouping elements, which can be repeatedly broken down until the actual items are reached. For example, in a Ranking custom group, the top-level element is Sales, which can be separated into the bands of Top Cities, Average Cities, and Bottom Cities. Each band can be further divided into elements such as San Diego and Berlin. By default, only the element names are displayed on the report, as shown in the following sample.

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Metrics</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales</td>
<td></td>
<td>$191,039.00</td>
</tr>
<tr>
<td>Top Cities</td>
<td></td>
<td>$91,826.00</td>
</tr>
<tr>
<td>Average Cities</td>
<td></td>
<td>$62,377.00</td>
</tr>
<tr>
<td>Bottom Cities</td>
<td></td>
<td>$36,836.00</td>
</tr>
</tbody>
</table>

If you change the display option of the custom group element in the Custom Group Editor, this division can be displayed in more detail. For example, the following is the same report with the element names and individual items displayed.
While hierarchical display is similar to drilling on a report, drilling requires additional executions of the report. However, drilling allows you to manipulate a report on the fly.

To view custom groups in a hierarchical structure on a report, you must

- expand the item display of at least one custom group element
- enable hierarchical display for the custom group
- display the hierarchical view in the report
These tasks are completed at different levels and in different interfaces, as described in the following table.

<table>
<thead>
<tr>
<th>Level</th>
<th>Target</th>
<th>Interface</th>
<th>Settings</th>
</tr>
</thead>
</table>
| Element     | Individual custom group element | Display Options in Custom Group Editor | Display  
• only the element names  
• only the individual items within this element  
• the element names, individual items, and expand the items if possible  
• the individual items and expand them if possible  
**Note:** The last option is available only for banding. |
| Custom group | All elements of a custom group | Options in Custom Group Editor          | • Enable hierarchical display  
• Enable subtotals  
• Position of element headers |
| Report      | All custom groups on a report | Report Data Options in Report Editor     | • Display custom groups in hierarchical or flat view                       |
|             | Individual custom groups    | Report Data Options in Report Editor     | • Enable subtotals |
| Project     | All custom groups in a project | My Preferences in Desktop                | • Show advanced qualification in Custom Group Editor  
• Show all prompt buttons in Custom Group Editor  
• Show tip boxes in Custom Group Editor  
• Trim leading spaces |

For detailed instructions on setting the various options, see the online help.
Sorting custom groups

A custom group is a convenient method to group attributes at the display level only. By default, you cannot sort on custom groups on reports. You can sort a custom group in either of the following ways:

- **Sort on the IDs of the attributes** that compose the custom group.

- **Inherit the attribute sort**, which uses the default sort of the attribute set to display first. The display order is set in the Report display forms list on the Attribute Editor display tab. The default sort for each form is determined in the New Attribute Form dialog box.

For example, a custom group uses the Call Center attribute, which has both a numeric ID and a description. The description is set to display on reports, and its default sort is Descending. If Inherit the attribute sort is used, the custom group is sorted in reverse alphabetical order by the description, as shown below:

<table>
<thead>
<tr>
<th>Areas</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>Seattle</td>
<td>$ 434,199</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>$ 1,050,983</td>
</tr>
<tr>
<td></td>
<td>San Diego</td>
<td>$ 2,397,919</td>
</tr>
<tr>
<td></td>
<td>Salt Lake City</td>
<td>$ 418,415</td>
</tr>
<tr>
<td>East</td>
<td>Washington, DC</td>
<td>$ 1,413,865</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>$ 1,009,416</td>
</tr>
<tr>
<td></td>
<td>Miami</td>
<td>$ 933,170</td>
</tr>
<tr>
<td></td>
<td>Charleston</td>
<td>$ 1,996,475</td>
</tr>
<tr>
<td></td>
<td>Boston</td>
<td>$ 1,325,448</td>
</tr>
<tr>
<td></td>
<td>Atlanta</td>
<td>$ 1,083,016</td>
</tr>
<tr>
<td>Central</td>
<td>New Orleans</td>
<td>$ 867,240</td>
</tr>
<tr>
<td></td>
<td>Milwaukee</td>
<td>$ 1,340,391</td>
</tr>
<tr>
<td></td>
<td>Memphis</td>
<td>$ 513,751</td>
</tr>
<tr>
<td></td>
<td>Fargo</td>
<td>$ 432,879</td>
</tr>
</tbody>
</table>
If ID sort is used, the custom group is sorted by the numeric ID of the attribute, as shown below:

<table>
<thead>
<tr>
<th>Areas</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>San Diego</td>
<td>$2,397,919</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>$1,050,983</td>
</tr>
<tr>
<td></td>
<td>Salt Lake City</td>
<td>$418,415</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>$434,199</td>
</tr>
<tr>
<td>East</td>
<td>Atlanta</td>
<td>$1,083,016</td>
</tr>
<tr>
<td></td>
<td>Washington, DC</td>
<td>$1,413,865</td>
</tr>
<tr>
<td></td>
<td>Miami</td>
<td>$933,170</td>
</tr>
<tr>
<td></td>
<td>Boston</td>
<td>$1,325,448</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>$1,009,416</td>
</tr>
<tr>
<td></td>
<td>Charleston</td>
<td>$1,999,475</td>
</tr>
<tr>
<td>Central</td>
<td>Milwaukee</td>
<td>$1,340,391</td>
</tr>
<tr>
<td></td>
<td>New Orleans</td>
<td>$867,240</td>
</tr>
<tr>
<td></td>
<td>Fargo</td>
<td>$432,879</td>
</tr>
<tr>
<td></td>
<td>Memphis</td>
<td>$513,751</td>
</tr>
</tbody>
</table>

**Sorting by metric values of items**

You can use the Keep Group Structure option to sort by the metric values of the items in each custom group element. For example, the Areas custom group used previously is set to display the individual items within its elements. It is placed on a report with the Revenue metric. Before sorting, the report is displayed as shown below:

<table>
<thead>
<tr>
<th>Areas</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>San Diego</td>
<td>$2,397,919</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>$1,050,983</td>
</tr>
<tr>
<td></td>
<td>Salt Lake City</td>
<td>$418,415</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>$434,199</td>
</tr>
<tr>
<td>East</td>
<td>Atlanta</td>
<td>$1,083,016</td>
</tr>
<tr>
<td></td>
<td>Washington, DC</td>
<td>$1,413,865</td>
</tr>
<tr>
<td></td>
<td>Miami</td>
<td>$933,170</td>
</tr>
<tr>
<td></td>
<td>Boston</td>
<td>$1,325,448</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>$1,009,416</td>
</tr>
<tr>
<td></td>
<td>Charleston</td>
<td>$1,999,475</td>
</tr>
<tr>
<td>Central</td>
<td>Milwaukee</td>
<td>$1,340,391</td>
</tr>
<tr>
<td></td>
<td>New Orleans</td>
<td>$867,240</td>
</tr>
<tr>
<td></td>
<td>Fargo</td>
<td>$432,879</td>
</tr>
<tr>
<td></td>
<td>Memphis</td>
<td>$513,751</td>
</tr>
</tbody>
</table>
To sort each Call Center (item) within each area (element), add the custom group as a sort and change the Criteria to Keep Group Structure. Add the metric as a second sort. The report now displays the call centers in order of their revenue values, within each element:

<table>
<thead>
<tr>
<th>Areas</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>Salt Lake City</td>
<td>$ 418,415</td>
</tr>
<tr>
<td></td>
<td>Seattle</td>
<td>$ 434,199</td>
</tr>
<tr>
<td></td>
<td>San Francisco</td>
<td>$ 1,050,983</td>
</tr>
<tr>
<td></td>
<td>San Diego</td>
<td>$ 2,397,919</td>
</tr>
<tr>
<td>East</td>
<td>Miami</td>
<td>$ 933,170</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>$ 1,009,416</td>
</tr>
<tr>
<td></td>
<td>Atlanta</td>
<td>$ 1,083,016</td>
</tr>
<tr>
<td></td>
<td>Boston</td>
<td>$ 1,325,448</td>
</tr>
<tr>
<td></td>
<td>Washington, DC</td>
<td>$ 1,413,865</td>
</tr>
<tr>
<td></td>
<td>Charleston</td>
<td>$ 1,995,475</td>
</tr>
<tr>
<td>Central</td>
<td>Fargo</td>
<td>$ 432,879</td>
</tr>
<tr>
<td></td>
<td>Memphis</td>
<td>$ 513,751</td>
</tr>
<tr>
<td></td>
<td>New Orleans</td>
<td>$ 867,240</td>
</tr>
<tr>
<td></td>
<td>Milwaukee</td>
<td>$ 1,340,391</td>
</tr>
</tbody>
</table>

**Changing the position of totals**

You can change the position of the totals for the custom group. This option is enabled only if subtotals have been enabled for the custom group. You can choose to display the totals at the top or bottom position, or inherit the default custom group definition.
For example, the following report displays the Total Revenue on the top of the report:

<table>
<thead>
<tr>
<th>Areas</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td></td>
<td>$30,440,334</td>
</tr>
<tr>
<td>West</td>
<td></td>
<td>$4,301,516</td>
</tr>
<tr>
<td>San Diego</td>
<td></td>
<td>$2,397,919</td>
</tr>
<tr>
<td>San Francisco</td>
<td></td>
<td>$1,050,983</td>
</tr>
<tr>
<td>Salt Lake City</td>
<td></td>
<td>$418,415</td>
</tr>
<tr>
<td>Seattle</td>
<td></td>
<td>$434,199</td>
</tr>
<tr>
<td>East</td>
<td></td>
<td>$7,764,390</td>
</tr>
<tr>
<td>Atlanta</td>
<td></td>
<td>$1,083,016</td>
</tr>
<tr>
<td>Washington, DC</td>
<td></td>
<td>$1,413,865</td>
</tr>
<tr>
<td>Miami</td>
<td></td>
<td>$933,170</td>
</tr>
<tr>
<td>Boston</td>
<td></td>
<td>$1,325,448</td>
</tr>
<tr>
<td>New York</td>
<td></td>
<td>$1,009,416</td>
</tr>
<tr>
<td>Charleston</td>
<td></td>
<td>$1,999,475</td>
</tr>
<tr>
<td>Central</td>
<td></td>
<td>$3,154,261</td>
</tr>
<tr>
<td>Milwaukee</td>
<td></td>
<td>$1,340,391</td>
</tr>
<tr>
<td>New Orleans</td>
<td></td>
<td>$867,240</td>
</tr>
<tr>
<td>Fargo</td>
<td></td>
<td>$432,879</td>
</tr>
<tr>
<td>Memphis</td>
<td></td>
<td>$513,751</td>
</tr>
</tbody>
</table>

Changing the position of element headers

You can change the position of the element headers relative to its elements. You can choose to display the element headers at the top or bottom position, or inherit the default custom group definition. For this, you must set the display option of the custom group to display both the element headers and its elements. If the display option of the custom group is set to display only either the element header or the elements, you cannot change the position of the element headers.
For example, the following report displays the element headers below their respective elements:

<table>
<thead>
<tr>
<th>Areas</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Diego</td>
<td></td>
<td>$2,397,919</td>
</tr>
<tr>
<td>San Francisco</td>
<td></td>
<td>$1,050,983</td>
</tr>
<tr>
<td>Salt Lake City</td>
<td></td>
<td>$416,415</td>
</tr>
<tr>
<td>Seattle</td>
<td></td>
<td>$434,199</td>
</tr>
<tr>
<td>West</td>
<td></td>
<td>$4,301,516</td>
</tr>
<tr>
<td>Atlanta</td>
<td></td>
<td>$1,083,016</td>
</tr>
<tr>
<td>Washington, DC</td>
<td></td>
<td>$1,413,865</td>
</tr>
<tr>
<td>Miami</td>
<td></td>
<td>$933,170</td>
</tr>
<tr>
<td>Boston</td>
<td></td>
<td>$1,325,448</td>
</tr>
<tr>
<td>New York</td>
<td></td>
<td>$1,009,416</td>
</tr>
<tr>
<td>Charleston</td>
<td></td>
<td>$1,999,475</td>
</tr>
<tr>
<td>East</td>
<td></td>
<td>$7,764,390</td>
</tr>
<tr>
<td>Milwaukee</td>
<td></td>
<td>$1,340,391</td>
</tr>
<tr>
<td>New Orleans</td>
<td></td>
<td>$867,240</td>
</tr>
<tr>
<td>Fargo</td>
<td></td>
<td>$432,879</td>
</tr>
<tr>
<td>Memphis</td>
<td></td>
<td>$513,751</td>
</tr>
<tr>
<td>Central</td>
<td></td>
<td>$3,154,261</td>
</tr>
</tbody>
</table>

**Custom groups and SQL**

You can also regard custom groups as many different reports just “stacked up” together. The SQL for a report with a custom group is likely to be very complex. Each of the individual “mini-reports” that make up the entire custom group report will have at least one, perhaps more SQL passes of its own. The Analytical Engine stacks up all these “mini-reports” to create the final results. In addition, numerous temporary tables may be created and dropped to hold intermediate data.

Therefore, running a report with a custom group is equivalent to running many different reports and putting them together. As a result, custom groups are SQL-intensive in the sense that they are likely to generate many passes of SQL to the database.
Example: custom groups

Report requirements

After completing an inventory report for the past six months, you notice that you have an excess of certain items in your warehouse. As a special promotion, you would like to offer these items at discounted prices to your best customers. To do this, you need to obtain a list of your top ten customers along with a list of your five lowest selling items on the same report.

How can you accomplish this?

Solution

You are really asking for two different reports: the top ten customers and the five lowest selling inventory items. You can create a custom group with the following custom group elements:

- top ten customers
- five lowest selling items

Each element will have a different qualification applied to it. In this case, the first element is the top ten customers ranked by dollar sales. The second element is the bottom five items by dollar sales. For each element, change the display options to show the element names and individual items. This allows the names of the customers and items to be displayed.

Create a report with this custom group and run it. Your top ten customers and bottom 5 items, in terms of revenue, are displayed.
Consolidations

Consolidations enable you to group together and to pick specific attribute elements. Further, consolidations allow you to place this grouping of attribute elements on a template just like an attribute. The elements of the consolidation appear as rows on your report, and they can have arithmetic calculations.

For example, suppose you wanted to see each season as a separate row on a report, but Season does not exist as an attribute in your project. A consolidation allows you to group together the elements of the Month of Year attribute into various seasons and place them on the template. This consolidation will contain four consolidation elements, one for each season.

Summer consists of June + July + August, fall consists of September + October + November, and so on. The consolidation is placed in the rows of your report with the desired metrics in the columns. Therefore, when a user runs the report, the metric values for June, July, and August are added together to yield the value for Summer. This occurs for each of the seasons.

Consolidation elements do not have to be based on a single attribute, as described in this example. You can use attributes at different levels, such as Region and Country, or unrelated attributes, such as Country and Year. For more information, see Consolidation elements.

In general, consolidations provide two powerful functions that enhance your reporting needs. These two functions are

- create a “virtual” attribute
- perform row level math
Create a “virtual” attribute

In the above example of the Seasons consolidation, the four different season consolidation elements are made up by adding together the respective Months of Year that belong to the different seasons. The fact that you can add together attribute elements in groups means that you can aggregate data on a report at a level other than one of the predefined attributes. The effect appears as if you had a Seasons attribute in your data model, as shown below.

<table>
<thead>
<tr>
<th>Seasons</th>
<th>Year</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summer</td>
<td>2002</td>
<td></td>
<td>$1,784,523</td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td></td>
<td>$1,912,180</td>
</tr>
<tr>
<td>Winter</td>
<td>2002</td>
<td></td>
<td>$2,539,620</td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td></td>
<td>$2,003,397</td>
</tr>
<tr>
<td>Fall</td>
<td>2002</td>
<td></td>
<td>$1,947,487</td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td></td>
<td>$2,102,240</td>
</tr>
<tr>
<td>Spring</td>
<td>2002</td>
<td></td>
<td>$2,605,000</td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td></td>
<td>$2,041,987</td>
</tr>
</tbody>
</table>

Of course, you can get the same effect if you change your data model, and actually add a Seasons attribute to your Time hierarchy. However, adding an attribute is generally a very complex task because you have to ensure that the proper lookup and relationship tables exist in the warehouse. Consolidations allow you to avoid changing the data model, although in a limited way.

This Seasons consolidation is built by adding together respective Months of Year in the different seasons. But you are not limited to just adding. In fact, you can perform any simple arithmetic operation while building your consolidation.
Perform row level math

Consolidations allow mathematical operations between elements or element groups. That is, you can perform arithmetic operations such as addition, multiplication, division, and subtraction. You can even use constants while specifying your elements.

This feature makes consolidations a powerful tool in reporting. It allows you to specify row level math for a report. That is, it allows you to have a row in a report that is specified by a mathematical operation.

Continuing with the Seasons example, the ratio of sales between different seasons can be calculated using row level math in a consolidation, as shown below:

<table>
<thead>
<tr>
<th>Seasons</th>
<th>Year</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summer</td>
<td>2002</td>
<td>$1,784,523</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td>$1,912,180</td>
<td></td>
</tr>
<tr>
<td>Winter</td>
<td>2002</td>
<td>$2,536,620</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td>$2,003,397</td>
<td></td>
</tr>
<tr>
<td>Fall</td>
<td>2002</td>
<td>$1,947,487</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td>$2,102,240</td>
<td></td>
</tr>
<tr>
<td>Spring</td>
<td>2002</td>
<td>$2,605,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td>$2,041,987</td>
<td></td>
</tr>
<tr>
<td>Spring/Fall</td>
<td>2002</td>
<td>$1.34</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td>$0.97</td>
<td></td>
</tr>
<tr>
<td>Summer/Winter</td>
<td>2002</td>
<td>$0.70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2003</td>
<td>$0.95</td>
<td></td>
</tr>
</tbody>
</table>

Spring/Fall is the consolidation element Spring divided by the consolidation element Fall. Similarly, Summer/Winter is the consolidation element Summer divided by the consolidation element Winter. The Seasons consolidation performs row level math that makes this report possible. Without consolidations, creating this analysis would be cumbersome.

Notice that elements are formatted as dollars ($) and percentages (%) in the same consolidation. You can format individual consolidation elements in the Consolidation Editor.
**Consolidation elements**

*Consolidation elements* are attribute elements that define the consolidation. Consolidation elements can also be an expression of attribute elements that make up a consolidation. They can be defined from any of the following:

- elements of the same attribute, such as two cities
- attribute elements from different levels, such as Region and Calling Center
- elements from unrelated attributes, such as Country and Year
- existing consolidation elements, such as the ratio of Spring and Summer sales to Fall and Winter sales
- elements from any other consolidation in the project, that is, elements imported from an existing consolidation into another one

You can combine the elements with simple mathematical expressions. For example, you can have an expression that adds attribute elements together, such as combining June, July, and August to get a Summer consolidation element. A consolidation element can also contain the logical operator AND. The following example demonstrates the use of mathematical expressions (addition and subtraction) and the AND operator.

**Example of AND used in a consolidation element expression**

You must report on the difference in revenues between the USA and Web for the winter of 2002. Create the following consolidation elements:

- USA: Winter 2002

\[
\{\text{Month}=\text{Jan }02 \text{ AND Country=USA}\} + \{\text{Month}=\text{Feb }02 \text{ AND Country=USA}\} + \{\text{Month}=\text{Mar }02 \text{ AND Country=USA}\}
\]
• Web: Winter 2002

  ({Month=Jan 02 AND Country=Web} + {Month=Feb 02 AND Country=Web} + {Month=Mar 02 AND Country=Web})

You cannot type AND into the expression. You must drag and drop an attribute element into the expression to trigger the AND operator. For more details, see the online help.

Create a consolidation element that uses the above elements to calculate the difference:

• USA - Web: Winter 2002


Finally, create a report with this consolidation and the Revenue metric. The report looks like the following:

<table>
<thead>
<tr>
<th>Winter 2002: USA &amp; Web</th>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA: Winter 2002</td>
<td></td>
<td>$1,715,103</td>
</tr>
<tr>
<td>Web: Winter 2002</td>
<td></td>
<td>$163,341</td>
</tr>
<tr>
<td>USA - Web: Winter 2002</td>
<td></td>
<td>$1,547,762</td>
</tr>
</tbody>
</table>

**Elements of the same attribute**

A consolidation can contain elements of the same attribute, such as (March) and (April), both elements of the attribute Month of Year. With reference to the previous example, consolidation elements allow you to expand the consolidation to see the values for each month. For example, using elements of the same attribute, you can modify the report result set as follows by adding the following three elements to the consolidation:

• Element 1 (March)
  
  Month of Year=March

• Element 2 (April)
  
  Month of Year=April
With the use of consolidation elements, the report can now display the following.

<table>
<thead>
<tr>
<th>Top Months</th>
<th>Metrics</th>
<th>Dollar Sales</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>March</td>
<td></td>
<td>$27,545.00</td>
<td>$4,940</td>
</tr>
<tr>
<td>April</td>
<td></td>
<td>$11,502.00</td>
<td>$1,957</td>
</tr>
<tr>
<td>March-April</td>
<td></td>
<td>$16,043.00</td>
<td>$2,983</td>
</tr>
</tbody>
</table>

A consolidation can contain any expression on the pairs of elements, such as (March - April). Using another example, an element expression can also be [DC, 2002] / [DC, 2003].

### Elements from different levels

A consolidation can contain elements from different levels within the same hierarchy, such as Item and Subcategory from the Products hierarchy. For example, you may want to compare the contribution of different items to the Subcategory sales. Your consolidation, for the items Chocolate Roses and Chocolate Spoons, looks like:

- **Element 1 (Roses percent)**
  
  `{Item=Chocolate Roses} / {Subcategory=Chocolate]`

- **Element 2 (Spoons percent)**
  
  `{Item=Chocolate Spoons} / {Subcategory=Chocolate]`

With the use of consolidation elements, the report displays the following.

<table>
<thead>
<tr>
<th>Chocolate percentages</th>
<th>Metrics</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roses percent</td>
<td>2.13%</td>
<td></td>
</tr>
<tr>
<td>Spoons percent</td>
<td>10.66%</td>
<td></td>
</tr>
</tbody>
</table>
Elements from unrelated attributes

A consolidation element can contain elements from different attributes. For example, you can calculate the difference between two different regions for a particular month. For the months March and April, the consolidation could contain the following elements:

- **Element 1 (March Southeast - Southwest)**
  \[\text{Month of Year}=\text{March AND Region}=\text{South-East}\]  
  - \[\text{Month of Year}=\text{March AND Region}=\text{South-West}\]

- **Element 2 (April Southeast - Southwest)**
  \[\text{Month of Year}=\text{April AND Region}=\text{South-East}\]  
  - \[\text{Month of Year}=\text{April AND Region}=\text{South-West}\]

The report now appears as follows:

<table>
<thead>
<tr>
<th>Region and Month</th>
<th>Metrics</th>
<th>Dollar Sales</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>March Southeast - Southwest</td>
<td>$ 1,659.00</td>
<td>$ 216</td>
<td></td>
</tr>
<tr>
<td>April Southeast - Southwest</td>
<td>$ 43.00</td>
<td>-$ 15</td>
<td></td>
</tr>
</tbody>
</table>

Existing elements

Using existing elements allows you to perform row level math, as described previously. For an example, see *Perform row level math*.

Importing elements

You can import consolidation elements from an existing consolidation. When a consolidation element is imported, a new consolidation element is created and embedded into the consolidation.
Evaluation order

If you want to place two or more consolidations on a report, the order the engine evaluates them is significant and can change your result set. If one of the consolidations involves multiplication or division and the other involves addition or subtraction, which consolidation is calculated first matters. When performing a mathematical calculation, the product of a sum is not always equal to the sum of the product.

For example, a report contains the Dollar Sales metric and two consolidations. One consolidation is Seasons, as discussed in the previous examples. The other is called Years and is composed of three elements: 2002, 2003, and 2002/2003. The row for Spring 2002/2003 can be calculated either as (March 2002 + April 2002 + May 2002) / (March 2003 + April 2003 + May 2003) or as (March 2002 / March 2003) + (April 2002 / April 2003) + (May 2002 / May 2003). When the first calculation is used, that is, the Seasons consolidation is evaluated first, the following report results.

<table>
<thead>
<tr>
<th>Years</th>
<th>Seasons</th>
<th>Metrics</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>Spring</td>
<td>$2,605,000.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Summer</td>
<td>$1,784,523.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fall</td>
<td>$1,947,497.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Winter</td>
<td>$2,539,620.00</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>Spring</td>
<td>$2,041,987.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Summer</td>
<td>$1,912,180.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fall</td>
<td>$2,102,240.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Winter</td>
<td>$2,003,397.00</td>
<td></td>
</tr>
<tr>
<td>2002/2003</td>
<td>Spring</td>
<td>$1.28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Summer</td>
<td>$0.93</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fall</td>
<td>$0.93</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Winter</td>
<td>$1.27</td>
<td></td>
</tr>
</tbody>
</table>
When the second calculation is used, that is, the Years consolidation is evaluated first, the following report results. Notice the difference in the 2002/2003 rows.

<table>
<thead>
<tr>
<th>Years</th>
<th>Seasons</th>
<th>Metrics</th>
<th>Dollar Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>Spring</td>
<td></td>
<td>$2,605,000.00</td>
</tr>
<tr>
<td></td>
<td>Summer</td>
<td></td>
<td>$1,784,523.00</td>
</tr>
<tr>
<td></td>
<td>Fall</td>
<td></td>
<td>$1,947,487.00</td>
</tr>
<tr>
<td></td>
<td>Winter</td>
<td></td>
<td>$2,539,620.00</td>
</tr>
<tr>
<td>2003</td>
<td>Spring</td>
<td></td>
<td>$2,041,987.00</td>
</tr>
<tr>
<td></td>
<td>Summer</td>
<td></td>
<td>$1,912,180.00</td>
</tr>
<tr>
<td></td>
<td>Fall</td>
<td></td>
<td>$2,102,240.00</td>
</tr>
<tr>
<td></td>
<td>Winter</td>
<td></td>
<td>$2,003,397.00</td>
</tr>
<tr>
<td>2002/2003</td>
<td>Spring</td>
<td></td>
<td>$3.97</td>
</tr>
<tr>
<td></td>
<td>Summer</td>
<td></td>
<td>$2.79</td>
</tr>
<tr>
<td></td>
<td>Fall</td>
<td></td>
<td>$2.83</td>
</tr>
<tr>
<td></td>
<td>Winter</td>
<td></td>
<td>$4.21</td>
</tr>
</tbody>
</table>

The evaluation order is set in the Report Data Options dialog box of the Report Editor when you create the report. To access this dialog box, select Report Data Options from the Data menu.

**Consolidations and SQL**

The calculations associated with a consolidation are done by the Analytical Engine component of the Intelligence Server. The SQL Engine writes the SQL query that gets the required data from the warehouse, and then passes it to the Analytical Engine to do any mathematical operation that is needed to create the report.

For example, the following SQL is for the dollar sales by season report discussed in *Create a “virtual” attribute*.

```
SELECT...
```

The seasons are not mentioned. The query retrieves the data for the Months of Year, and then the Analytical Engine performs the necessary calculations to present the data in terms of seasons.
select a12.[MONTH_OF_YEAR] AS MONTH_OF_YEAR, 
max(a13.[MONTH_OF_YEAR_NAME]) AS MONTH_OF_YEAR_NAME, a12.[YEAR_ID] AS YEAR_ID, 
sum(a11.[TOT_DOLLAR_SALES]) as DOLLARSALES 
from [MNTHCATEGORY_SLS] a11, [LU_MONTH] a12, 
[LU_MONTH_OF_YEAR] a13 
where a11.[MONTH_ID] = a12.[MONTH_ID] AND 
a12.[MONTH_OF_YEAR] = a13.[MONTH_OF_YEAR] AND 
a12.[MONTH_OF_YEAR] in (3, 4, 5, 6, 7, 9, 10, 11, 12, 1, 2) 
group by a12.[MONTH_OF_YEAR], a12.[YEAR_ID]

Example: consolidations

Report requirement

You have been given a task to understand how products are performing in different sections, or territories, of the country and abroad. This will allow you insight into consumer buying patterns and offer guidance on establishing pricing strategies and promotions. You will need to see the territories in the rows of your report and various metrics, such as sales, profit, and revenue in the columns.

How can you accomplish this?

Solution

A Territory attribute does not exist in your project. You will need to create one.
A consolidation allows you to group together various elements of the Region attribute into various territories and place them on your template. In this example, you will need to break down the territories as follows:

- East = Mid-Atlantic, South, South
- West = Central, Northwest, Southwest
- Foreign = Canada, England, France, Germany

These consolidations placed in the rows of your report allow the metrics for values to be added together for a specific territory. For example, the metric values for Central, Northwest, and Southwest will be added together to yield the value for West, and so on.

**Custom group and consolidation comparison**

Both consolidations and custom groups provide flexibility in reports, but the objects differ in their structure and use. The essential distinction is that consolidations work with attributes and custom groups use filters. Consolidations are groupings of attribute elements while custom groups are based on filter criteria. Custom groups are used to apply different filters to different rows of a report. Consolidations are used to create virtual attributes to allow reporting on attributes that do not exist in the data model. Finally, row level math can be performed with consolidations but not with custom groups.

Custom groups are more flexible than consolidations because you do not have to know much about your data to create filters for a custom group. In contrast, consolidations require that you know exactly which attribute elements to select when creating the consolidation. To continue with the examples from the previous sections, you create filters for the Store Inventory custom group, to group small stores with low inventory and large stores with low inventory. For the Seasons consolidations, you need to know the months that make up a season.
The following table outlines other differences between custom groups and consolidations. More information on each section follows the table.

<table>
<thead>
<tr>
<th>Arithmetic operations (row level math)</th>
<th>Custom Group</th>
<th>Consolidation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not allowed</td>
<td>Allowed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Site of final calculation</th>
<th>Warehouse</th>
<th>Analytical Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL efficiency</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Recursive definition</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Display mode</td>
<td>Flexible and expandable</td>
<td>Fixed at element level only</td>
</tr>
<tr>
<td>Subtotals</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

### Arithmetic operations

Arithmetic operations such as addition and division are not allowed in custom group definitions. However, complicated mathematical expressions can be created in consolidations using the following operators:

- +
- -
- *
- /
- ()

This means that row level math, or mathematical operations between elements, can be performed in consolidations but not in custom groups. Row level math is a powerful tool in reporting, since it allows rows to be specified by mathematical operations.
Site of final calculation

Although the Analytical Engine might be needed to resolve banding in a custom group, the final stage of calculation is always in the data warehouse. For consolidations, all necessary data is retrieved from the data warehouse and then the consolidations are created in the Analytical Engine. Therefore, consolidations can be assigned an evaluation order to provide more and varied analyses. For more information on evaluation order, refer to Evaluation order in this chapter.

SQL efficiency

For each custom group element, there is at least one SQL pass. When the custom group element is expanded in the Custom Group Editor, up to three SQL passes must be made for each custom group element.

Since the Query Engine uses a smart algorithm to combine consolidation elements and determine the minimal number of SQL passes, only one SQL pass may be needed for all the consolidation elements.

Recursive definition

You cannot use existing custom group elements to build a new custom group element. You must always build a custom group from attributes. In contrast, you can build new consolidation elements based on existing consolidation elements.
Display mode

Each custom group element can be viewed as a set of smaller grouping elements, which can be repeatedly broken down until the actual items are reached. This functionality, along with the four different display modes, provides flexibility and deeper analysis. For more information, see the Custom group options section in this chapter.

Consolidations are displayed only at the element level; you cannot expand the elements.

Subtotals

Custom groups act like attributes when subtotals are included.

Subtotals cannot be used with consolidations. You can, however, create a subtotal using a consolidation element. For example, if your consolidation contains the elements Spring and Summer, create another element that adds Spring and Summer.
Introduction

A prompt is a MicroStrategy object that allows user interaction at report run time. The prompt object is incomplete by design. The user is asked during the report resolution phase of report execution to provide an answer to complete the information. For example, the user can enter information such as the region “North East” or year “2002,” and the data is returned from the data warehouse. With prompts you can create reports that allow users to change the report content at run time.

Prompts are useful for asking questions about the set of data you would like to see in the report. Prompts allow the report to have dynamic report definitions, which can change with each query by altering the information in the prompt dialog box. Prompts can be a part of the report definition, filter, template, custom group, or metric.
There are benefits to using prompts as a part of these objects. Prompts are useful to keep the number of objects in a project small, as they allow you to define how you would like to view things in the report environment instead of providing a separate report for each option.

You will be able to find the report that you would like to see quicker because there will be fewer options. However, prompting questions asked of the user raise the complexity of just running a report, and you run the risk of confusion. This confusion can be allayed by providing good descriptions for the prompts so that the user is clear on what question they are answering.

This chapter explains the concepts necessary to create and use prompts.

What is a prompt?

By using a prompt in a filter or template, you can

- apply filtering conditions to the report, a custom group on the report, or a metric on the report
- choose what objects, such as attributes and metrics, are included in the report

Prompts allow you to determine how the report returns data from the data warehouse. Prompts save you time. Instead of building several different reports, a simple question can be asked just before the report is executed. There are many types of prompts, and each type allows a different type of question to be asked. These prompt types will be discussed individually later on in this chapter.

One type of question that can be asked is a question about what subset of data should be included on the report. For example, to see a report about sales in a particular geographical region, you can build three reports—Sales in West Region, Sales in Central Region, and Sales in East Region. Instead, you can create one report called “Sales, prompted on Region” that asks the user for the region to include.
A different type of question to ask is what attribute should be shown on the report. For example, Sales by Day, Week, Month, Quarter, or Year can be consolidated via prompting. This report asks the user by which time unit the report data should be displayed.

Prompts, regardless of what type they are, have the following common features:

- search ability
- properties

**Prompt search ability**

You can use search objects when creating a prompt.

For example, you could create an object that asks you to choose from a list of metrics whose name has “Dollar Sales” in it. When the prompt is executed, a search for all metrics with those words in the name is performed. The results of that search are used to generate the list of possible metrics from which to choose for the prompt.

**Prompt properties**

Although each of the prompt types available has distinct capabilities to provide a specific set of conditions, there are certain properties that all prompts share:

- Title identifies and differentiates the prompt.
- Description indicates to the end user the nature or purpose of the prompt.
- Default contains the default answer to the prompt, if one was specified.
- Maximum and minimum determine how many answers a user must/is allowed to choose (optional).
- Web options define how the prompt appears in MicroStrategy Web.
Types of prompts

Using the following prompt types, you can create a prompt for nearly every part of a report. It is important to remember that prompts can be used in many objects including reports, filters, metrics, and custom groups. All of these prompts will reveal themselves at report execution time, but the origin of the prompt can be within any of the objects.

- Filter definition prompt encompasses four different prompt types, all of which allow the user to define the filtering criteria: attributes in a hierarchy, attribute forms, attribute element lists, and metrics.
- Object prompt allows you to select which MicroStrategy objects to include in a report, such as attributes, metrics, custom groups and so on. Object prompts can either determine the definition of the report template or the report filter.
- Value prompt allows you to select a single value such as a date, a specific number, or a specific text string. The value chosen by the user is compared to metric or attribute element values, and thus determines the data viewed by the user.
- Level prompt allows you to specify the level of calculation for a metric.

Filter definition prompts

These prompts are used for qualifying on the value of attribute elements and metrics. The filters affected by these types of prompts can be in the report, in a filter (which in turn may be used in the conditionality of a metric in the report), or in an element of a custom group. Additional information on custom groups may be found in Chapter 8, Custom Groups and Consolidations.
Choose from all attributes in a hierarchy

This type of prompt is used to qualify on one or more attributes in one or more hierarchies. You are presented with at least one hierarchy and all of the attributes included in that hierarchy. You can qualify on one or more of those attributes by choosing an element list or by qualifying on a particular attribute form. The choices made are included in the filtering criteria in the report.

To define this type of prompt, you can do one of the following:

- Choose a particular hierarchy.
- Use the set of hierarchies resulting from a search for hierarchies.
- List all hierarchies available in the project.

If you choose to display more than one hierarchy, you can make qualifications from all hierarchies presented at report run time.

Qualify on an attribute

This is used to apply conditions or qualifications to an attribute form.

The user is presented with one or more attributes and may qualify on an element list or an attribute form of one of them.

To define an attribute qualification prompt, you can either

- choose a particular attribute
- present the user with a partial or complete list of attributes that are the result of a search for attributes available in the project
Choose from an attribute element list

This option is used to allow the user to choose from a list of attribute elements to be included in a filter or custom group. This list may be restricted, at prompt design time. This type of prompt can be used with any attribute in a project.

The list of elements from which the user may choose can be implemented by

- selecting all elements associated with an attribute
- providing a partial list of elements by applying a filter on all of the elements associated with an attribute
- providing a predefined list of elements from which the user can choose

Qualify on a metric

A metric qualification prompt allows a user to qualify on a metric. The user is presented with one or metrics, and may choose one on which to qualify.

The choice of metrics can be defined by

- specifying a single metric for run-time use
- specifying a search object to restrict the list of metrics from which the user can choose
Example: filter definition prompt

Report requirement

You need to create a report showing sales in the West, Central, and Eastern Regions. All other data on the report remains the same. You do not necessarily want the regions on the same report.

Solution

To meet this requirement, the easiest solution is to create a report that includes a filter prompting the user on region. When the report is executed, the prompt dialog opens, asking the user to choose the region(s) for which to return the report results.

Object prompts

Object prompts are used to allow you to choose what objects will be included in the report filter or on the report template.

These are defined by specifying either a search object or a predefined list of objects from which the user can choose. An object prompt allows specification of default answers as well as maximum and minimum number of objects to be selected.

A search object defines the criteria (such as location, date, owner, and so on) for a list of objects to be generated. Searches defined in prompts are saved in the project. For example, a search object can display all metrics that are contained in a certain folder and use a particular fact.

All objects returned by a single object prompt must be of the same object type. For example, you can use a prompt to ask for metrics or attributes, but not for both. If you want to prompt for multiple object types in the same report, you may create an object prompt for each object type.
Example: object prompt

**Report requirement**

Create a report displaying item sales. At runtime, the user can select whether to calculate sales at the Category or the Subcategory level.

**Solution**

Create an object prompt with Category and Subcategory. Create the Sales metric, using the object prompt as the target of the level (dimensionality). When the report is executed, the user is prompted to select either Category or Subcategory. The Sales metric is calculated accordingly.

**Report requirement**

The Sales Manager frequently asks her analysts to provide similar reports with minor changes to the metrics. She always wants a metric to calculate Dollar Sales for each employee. In addition, she sometimes wants to compare the results of each employee to the sales results of the best or the worst employee. Other times, she wants to compare the results of each employee to the average sales of all the employees.

**Solution**

Rather than create many different reports, you would like to provide the Sales Manager with the flexibility to select which analytical function she wishes at the time of running the report.
In this case you can give her three functions to choose from:

- minimum
- maximum
- average

She will select which function to use with the Dollar Sales metric. Your final report can then have the following objects:

- employee
- Dollar Sales metric
- Dollar Sales metric that uses the analytical function selected by the user

**Value prompts**

Value prompts are used when the information desired at run time is a single value of a specific data type. The value chosen by the user is compared with either an attribute form or a metric. This comparison can be done in a filtering criteria or in a custom group. The different types of Value prompts are:

- **Date** - prompts for a date value.
- **Numeric** - prompts for a numeric value. Numeric value prompts accept integers or decimals up to 15 digits of precision.
- **Text** - prompts for any type of text.
- **Big Decimal** - prompts for a big decimal value. Big Decimal value prompts accept integers and decimals up to 38 digits of precision.

Big Decimal prompts should only be used in expressions that require high precision, such as qualifying on a Big Decimal attribute ID.

- **Long** - prompts for a long integer value. Long prompts accept integer numbers up to 9-10 digits.
Although long prompts are not part of the options available by default for selection, you can enable them as part of your project preferences.

Value prompts allow specification of maximum and minimum values to be applied.

**Example: value prompt**

**Report requirement**

Create a report showing sales since a certain date.

**Solution**

Prompt the user for the date since they want to see sales data. The value they choose is applied to a filter criteria for the attribute “date.” The prompt here is included in the filter on a report.

**Level prompts**

Level prompts are used to define the dimensionality of a metric. When two or more metrics differ only in level, it is useful to create a prompt on the dimensionality to avoid having to create two or more metrics.

This prompt definition type requires either a hierarchy or a list of attributes. The default output of the metric is at the report level.
Example: level prompt

Report requirement

Create a report listing items with sales calculated at either the Category or the Subcategory level. At runtime, the user must be able to select the Category or Subcategory level, as well as the filtering and grouping properties of the metric.

Solution

Create a level prompt with Category and Subcategory. Create the Sales metric, using the level prompt as the target of the level (dimensionality). When the report is executed, the user is prompted to specify the complete level of the metric, including the target, filtering, and grouping. The Sales metric is then calculated accordingly.

This example differs from the one in Example: object prompt. An object prompt allows prompting only on the target, while the level prompt allows prompting on all components of the metric—target, filtering, and grouping.

Saving reports with prompts

When you save a prompted report you have already executed, the Save Options dialog box opens. You have the following options:

- Static—You are not be prompted when you run the report again. The answers you chose the first time are used from now on.

- Prompts—You are prompted when you run the report again. Your choices include filter prompts, template prompts, or both. Information on filter prompts and template prompts can be found in Chapter 5, Filters.
After you execute a prompted report, you can make modifications to the report such as, adding a report object or changing the layout of the report. You can choose to save or clear these modifications on the report template by selecting the check box **When saving a report to be reprompted, rerun prompts included in objects in the report definition**. You can access this check box from **Report Data Options, General, Advanced** tab. For more information, see the online help.

**Example: basic prompts**

**Report requirement**

You need to run a series of reports, each showing the yearly sales for one of five non-consecutive years.

**Solution**

You can create a filter prompt on Year. When the user runs the report with this prompt, they are asked to specify which year he would like to see. The user can run the report the first time using 1998, and run it a second time using 2003, and so on.

**System prompts**

System prompts are built-in prompts that are created when the project is created. System Prompts are created in the System prompts sub folder in the Prompts folder of MicroStrategy Desktop.
The User Login system prompt is a special prompt that returns the current user's user login. The user is not prompted to provide an answer to the User Login prompt, as the Intelligence Server automatically answers the prompt with the user's login when prompts are resolved. The User Login system prompt can be used in conditions in which a text value prompt is used in a report.
Facts are one of the main building blocks for creating a project. A fact has two characteristics: it is numerical and aggregatable. Examples of facts may include sales dollars, units sold, profit, cost, and so on.

This chapter discusses advanced facts. It assumes that you already have a data warehouse populated with your data.

What is a fact?

Facts are objects created by and shared between MicroStrategy users. They relate numeric data values from the data warehouse to the MicroStrategy reporting environment. The facts you create allow users to access data stored in a data warehouse. Facts form the basis for metrics that are used in the majority of analyses and reports that users can create with MicroStrategy.
Facts are stored in the data warehouse in *fact tables*. These fact tables are composed of different columns, each cell representing a specific piece of information. This information is used to create a metric such as profit, which is a business measure.

Facts are based on physical columns in the data warehouse. When fact information is requested for a report in MicroStrategy, that column is accessed to retrieve the necessary data.

Like other schema objects such as attributes, facts are logical MicroStrategy objects that correspond to physical columns and tables. Unlike attributes, facts do not actually describe data. Facts are the actual data values stored at a specific fact level. A fact *entry level* is the lowest set of attributes at which a fact is stored.

Facts and attributes are necessary to define projects. In a MicroStrategy project, facts are numeric data and attributes are contextual data for the facts. As the project designer, you create projects that contain facts and attributes, which the users can include when building metrics and reports.

**Fact structure**

Every fact has the following:

- The Fact Definition is composed of one or more fact expressions. Every fact must have at least one expression.

- The Column Alias stores the column name, which MicroStrategy uses to generate SQL statements when creating temporary tables related to the fact. Every fact must have a column alias, and MicroStrategy selects a default column alias depending on the type of fact, unless you create a new column alias.
• Level Extensions (Advanced) allow facts stored in the data warehouse at one level to be reported at an unrelated level. You can also use extensions to prevent a specific fact from being reported at a certain level, even though it is stored at that level. Level extensions are not commonly applied but are very effective for special data modeling scenarios if you are an advanced user.

Note the following:

– For a fact to exist in a MicroStrategy project, both the fact expression and column alias must be defined by the product.

– During project creation, when you select the numeric column used to represent the fact, both the fact definition and column alias are automatically defined.

Fact definition

A fact definition contains properties that define a fact and its components. The fact definition is composed of at least one fact expression and basic information about the fact, including the fact name, expression, and the source table it uses.

The example below demonstrates a fact definition that includes Name, Description, and Expressions. Multiple expressions can exist within each definition.

The fact expressions contained in the definition represent how a fact is calculated by MicroStrategy. Facts can be found in multiple tables in a warehouse schema and often must be calculated differently from one table to the next.
Note the following:

- Each fact expression relates to one or more related tables that contain the fact.
- Fact expressions define, for each of the tables, how the fact is calculated.

**Fact expressions**

A fact expression can be as simple as a fact column name from the warehouse or as sophisticated as a formula containing fact columns and numeric constants. A fact definition can have one or more fact expressions.

A fact can be defined using an ApplySimple function. Apply functions are discussed in Appendix C, *Pass-through Expressions*.

The following illustrates a column in the fact table and the associated fact expressions.

Valid expressions are formulas constructed from fact columns with or without numeric constants or mathematical operators. The mathematical operators that can be used in an expression are

- addition (+)
- subtraction (-)
- multiplication (*)
- division (/)
**Implicit fact expressions**

Implicit facts are virtual or constant facts that do not physically exist in the database.

The implicit fact can have its expression defined as a constant value, though nothing is saved in a column. An implicit fact indicates a fact table from which to retrieve data.

**Derived fact expressions**

A derived fact has its value determined by an expression that contains more than just a column in a table. Any operation on a column, such as adding a constant, adding another column, or setting the expression to be an absolute value, creates a derived fact. In other words, you are making a new fact from information that is already there, as in the following example.

You have a table in your data warehouse, which contains the following elements:

<table>
<thead>
<tr>
<th>Fact Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
</tr>
<tr>
<td>Quarter</td>
</tr>
<tr>
<td>Quantity_Sold</td>
</tr>
<tr>
<td>Price</td>
</tr>
</tbody>
</table>

You can create a new fact, Sales, by creating this derived fact:

\[ \text{Sales} = \text{Quantity} \_ \text{Sold} \times \text{Price} \]

The advantage of creating a derived fact is that you do not have to create multiple facts. It hides the structure of the warehouse from your users and one consistent fact exists in the project in lieu of many facts being retrieved from multiple tables. For example, if you use the same formula in many places, it uses one pass of SQL instead of three.
Heterogeneous column names for facts

MicroStrategy allows you to identify heterogeneous fact column names for each fact. With **heterogeneous column names**, you can refer to the same fact with multiple column names from different tables that identify the same quantitative value. In the warehouse, the same fact can have different column names. You can use the Fact Editor to create fact expressions.

An example is dollar sales. The warehouse has two tables, as illustrated below. Table 1 contains a fact called Dollar_Sales. Table 2 includes a fact called Dollar_Sls. These two items represent the same information. By creating an heterogeneous fact column name, the system knows that these are the same fact when you call the information.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>Month</td>
</tr>
<tr>
<td>Dollar_Sales</td>
<td>Dollar_Sls</td>
</tr>
</tbody>
</table>

Column alias

A column alias specifies both the name of the column to be used in temporary tables and the datatype to be used for the fact. By default, the datatype for a fact is inherited from the datatype of the column on which the fact is defined. However, there are cases where you may need to change this.

For example, you can define a fact to be the difference between two dates, perhaps to perform a calculation such as the average number of days between a start and an end date. You could create this fact using the following expression:

```sql
ApplySimple("DateDiff(day,#0, #1)", [Start_Date_Id], [End_Date_Id])
```
The expression syntax is specific to your database type. This syntax is specific to Microsoft SQL Server. The SQL you create may be different.

The datatype for this fact is automatically set to a date datatype because the `Start_Date_ID` and `End_Date_ID` have date data types. However, the result of the calculation, that is, the difference of the two dates, is an integer.

This is used when a temporary SQL table needs to be created for the calculation. If you did not change the datatype of the column alias, then the system uses a date datatype and tries to insert integer data into this column. While this may not be a problem for some database platforms, it can cause an error. To avoid the possibility of an error due to conflicting data types, you should modify the column alias for the fact to change the default date datatype to an integer datatype.

### Level extensions

Facts are stored at a particular level in the warehouse. The fact level is defined by the attribute IDs present in the table. Level extensions are necessary when facts are stored in the data warehouse at one level and reported at a different level. Every fact is tied to a set of attributes that may or may not satisfy all user-level reporting requirements. An explicit fact extension is needed when a fact does not relate to all needed report attributes.

If a fact has an attribute in the entry level, all parent attributes are available for use as well, without extensions.
You can use level extensions to change a fact level. Level extensions define how facts can be extended, lowered, or disallowed to other attributes across the schema. By creating a level extension, you are allowing facts or attributes that have been captured at one level to be extended to other, technically unrelated levels for reporting reasons.

Level extensions are not requirements like the fact definition and column alias, and they tend to be used only in special cases.

Before a metric containing this fact can be used with an attribute that is not in or related to the entry level, the level extension must be defined. If the attribute is not related to the entry level, it must be above the fact entry to allow for degradations. You can accomplish this through one of the methods listed below:

- table relation
- fact relation
- degradation
- cross-product join
- disallow the fact entry level

These methods are defined through the Level Extension Wizard.
Table relation

A table relation defines a join on tables. When you create a join between a lookup or fact table, you are creating a table relation to extend a fact. A fact extension can be used to relate a fact to an attribute using a fact table. The join is important as the table contains an attribute in the entry level and the attribute to which to extend.

If you want to extend the fact so that it can be reported at any level in the hierarchy, you should choose the lowest level attribute in that hierarchy.

For example, your warehouse contains the following tables.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution Center</td>
<td>Distribution Center</td>
</tr>
<tr>
<td>Order</td>
<td>Order</td>
</tr>
<tr>
<td>Freight</td>
<td>Customer</td>
</tr>
<tr>
<td>Order Unit Sales</td>
<td></td>
</tr>
</tbody>
</table>

The entry level of the Freight fact is Distribution Center and Order. You want to define a template containing both Customer and Freight even when Freight is not captured at the level of Customer. You can relate Freight with Customer by joining Table 1 with Table 2 on the Distribution Center and Order attributes. To do this, define a table relation on the Freight fact using Table 2 so that the entry level of Freight becomes Distribution Center, Order, and Customer. This relationship can be denoted as:

Freight ———> Customer

| Table 2 on Distribution Center/Order |
When the engine processes a report containing Customer and Freight, it joins Table 1 and Table 2 and considers the resulting table as one logical fact table at the Distribution Center/Order/Customer level. The SQL generated for a report containing Distribution Center, Customer, and Freight is:

```sql
select a1.DIST_CENTER, a2.CUSTOMER,
       sum(a1.FREIGHT)
from TABLE1 a1, TABLE2 a2
where a1.DIST_CENTER = a2.DIST_CENTER and
      a1.ORDER = a2.ORDER
group by a1.DIST_CENTER, a2.CUSTOMER
```

**Fact relation**

Fact extensions can be defined by a fact relation instead of a table relation. With a fact relation, the table join is possible on any table that contains the fact. This allows more flexibility in defining the relations, since the MicroStrategy Engine is responsible for choosing the appropriate table to join.

The following diagram shows the schema from the table relations example after two summary tables are added to it.
To extend the entry level of the Freight fact to Customer, you can use a fact relation using the Order Unit Sales fact. The MicroStrategy Engine tries to join a table containing Freight to a table containing Order Unit Sales. The engine can make the following joins, depending on the join attributes specified:

- Join 1—Table 1 and Table 2 on Distribution Center, Order
- Join 2—Table 1 and Table 4 on Distribution Center
- Join 3—Table 2 and Table 3 on Distribution Center
- Join 4—Table 3 and Table 4 on Distribution Center
The fact relationship using Order Unit Sales can be denoted as the following.

```
Freight       Customer
       ↓
Order Unit Sales on Distribution Center/Order or Distribution Center
```

The join attributes can be either Distribution Center and Order or just Distribution Center. In the first case, only Join 1 is valid. In the second case, Joins 2, 3, and 4 are valid. The engine will choose the appropriate join.

The SQL generated for a report containing Distribution Center, Customer, and Freight is shown below, if the join attribute is Distribution Center.

```sql
select a1.DIST_CENTER, a2.CUSTOMER, 
    sum(a1.Freight)
from TABLE3 a1, TABLE4 a2
where a1.DIST_CENTER = a2.DIST_CENTER
group by a1.DIST_CENTER, a2.CUSTOMER
```

As with table relations, you can specify the best fit as the join strategy so that the engine calculates the joins. In a best fit join, the set of join attributes must contain the entire key of the left-hand-side fact table.

**Degradation**

*Degradation*, which lowers a fact level, is the logical opposite of aggregation. When facts exist at a higher level than the report display level, you must specify how the engine degrades the data to the lower level. When you lower the level at which a fact is reported, you are using degradation.
For example, if your fact is stored at the yearly level and you want to report the data at the monthly level, create a degradation. You also add an allocation expression to change the definition of the fact in a level extension. In this particular example, you select Month to be the attribute to which to degrade, and then specify that the allocation expression is fact/12. By creating allocation expressions, you are defining how higher-level facts are degraded to lower-level attributes. Allocation expressions are defined by operations you set on attributes and facts in the Level Extension Wizard.

**Cross product join**

You can use a cross product join when a join does not exist, and you need to force a fact to relate to an attribute by extending the fact. The cross product join allows a single fact value to relate to all elements of an unrelated attribute. This method can produce incorrect data because data can be repeated and counted twice in some cases. Cross products should only be used when no other way to extend the fact exists.

When you specify a cross product join to relate a fact to an attribute, you are creating a Cartesian product of the lookup attribute. As this method can be inefficient, MicroStrategy does not recommend using the cross product extension.

For example, in the following schema, Distribution Center does not relate to Dollar Sales.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution Center</td>
<td>Order Customer</td>
</tr>
<tr>
<td></td>
<td>Dollar Sales</td>
</tr>
</tbody>
</table>
To report Dollar Sales by Distribution Center, a cross product join must be used. The relation is defined as follows.

Dollar Sales → Distribution Center

Cross product

Notice that no join attributes are specified. The MicroStrategy Engine always cross-joins the lookup tables of the attributes in the extension.

The SQL generated for a report containing Customer, Distribution Center, and Dollar Sales is

```sql
select a1.DIST_CENTER, a2.CUSTOMER,
       sum(a2.DOLLAR_SALES)
from TABLE1 a1, TABLE2 a2
group by a1.DIST_CENTER
```

**Disallow the fact entry level**

The **Disallow partially or completely the fact entry level** setting is like a lock, which you use to prevent a fact from being reported at a specific level. The **Disallow partially or completely the fact entry level** setting prevents unnecessary joins to lookup tables. For example, your project has a fact at the Item level in the Product dimension. You can disallow an extension to an attribute, Day, so that it is not reported in the Time dimension.

The Disallow the fact entry level setting applies only to attributes that can be considered as extended attributes. For example, you create a report that contains the attributes Subcategory and Item and the Revenue metric, which is defined as sum of the fact Revenue. You now disallow an extension on the fact
Revenue for the Item attribute and update the schema. If you re-execute the report, you can still see Revenue by Item. This implies that the fact extension has not been disallowed. This is because Revenue exists at the same level as Item in the MicroStrategy tutorial. So you encounter only normal joins and no extensions.

Disallow fact extension examples:

1. If a fact is stored at a level that is counterproductive to a query, such as data that is stored at the Week, Day, Minute, or Second level, you can disallow the lower levels. For example, if you have three years’ worth of data, querying at the Minute or Second level consumes too many resources and returns extensive data. With a disallow in place, if you create a report and attempt to include the fact at the Minute or Second level, an error is returned, indicating that the report cannot be run at that level.

2. Consider a schema containing three dimensions—Geography, Time, and Product. Create a fact called Sales at the Item level in the Product dimension. Create a metric called Sales as the sum of the Sales fact. Now, create a report containing the Month attribute and the Sales metric. The analytical engine does a dynamic cross-join and evaluates this report. To explicitly disallow an extension of the Sales fact to the Time dimension, use the Disallow partially or completely the fact entry level setting and select the lowest attribute in the Time dimension, say, Day. Update the schema. When you re-execute the report, the report fails because the disallow setting now prevents the cross-joins between the lookup tables and fact tables. This setting, however, does not affect normal joins.

Use of partial disallow with contradictory extension definitions: In example 2 above, for the Sales fact, assume you specify an extension to the Month attribute and also disallow extension to Year which is a parent of the extended attribute, Month. If you execute the report containing the Year attribute and Sales metric, the report runs. In this case, the
engine sorts the extension conditions specified in some order and calculates the report based on the sorted order of extensions. This is not an expected design condition, although the engine returns a valid SQL. It is advisable to avoid fact definitions that contain contradictory extension definitions.

## Defining facts

The tools to create facts are the Fact Creation Wizard and the Fact Editor. Both of these tools are found in MicroStrategy Architect, which can be accessed through MicroStrategy Desktop. The Fact Creation Wizard is a step-by-step interface that is typically used when you first create a project. It allows you to create facts in bulk; that is, many facts in one creation process. The Fact Editor is used to add advanced features to facts that already exist or to add new facts during the project evolution.

It is important to understand the concept of how to define facts because facts are the basis for almost all metrics. They also allow you to create advanced metrics containing data that is not stored in the warehouse but can only be derived by extending facts.

### Example: fact definition

The following table lists fact definitions for simple facts.

<table>
<thead>
<tr>
<th>Fact name</th>
<th>Fact description</th>
<th>Expression</th>
<th>Fact level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Sales</td>
<td>Daily store item sales totals</td>
<td>Total_Sales</td>
<td>Item/Store/Day</td>
</tr>
<tr>
<td>Unit Price</td>
<td>Item prices, recorded on a daily basis</td>
<td>All_Sales</td>
<td>Item/Day</td>
</tr>
<tr>
<td>Inventory</td>
<td>Daily store item stock quantities</td>
<td>Item_Inv</td>
<td>Item/Store/Day</td>
</tr>
</tbody>
</table>
**Attributes**

**Introduction**

An attribute is mainly used to group or aggregate fact data.

You can group related attributes such as City, State, or Region, into a common hierarchy, like Location. In a logical data model, when attributes are in the same hierarchy they must be related to each other, whereas attributes in different hierarchies cannot be related.

This chapter describes what an attribute is, the different types of attributes, and the concepts necessary to use attribute elements, forms, and form expressions.
What is an attribute?

Attributes represent entities in the business model and are normally identified by a unique ID column in the data warehouse. The attribute acts like a column header and the data that appears in the following table are elements. Elements define and make up the attribute. For example, the elements New York, Baltimore, and Boston are grouped under the attribute City.

You can use attributes to define the level of detail for a report. The lowest level attribute you include in a report is the lowest level of detail reported. A high-level report, such as a report at the Year level, includes the Year attribute but lacks the detail of a similar report which includes the lower level attributes Month and Week. It is important to understand the data is still the same, although it is just not aggregated.

The following diagram shows attributes and elements as they display in a table.

Attributes are defined by these properties:

- Form contains an identifier or descriptor of an attribute, such as an abbreviation or URL.
- Expression maps a MicroStrategy attribute form to one or more columns in the warehouse.
- Relationship allows interaction of data and shows how data is related within a project.
The following diagram illustrates how the attribute properties are related.

**Attribute elements**

*Attribute elements* are the unique values or contents of an attribute. For example, if City is the attribute, then Chicago and Miami are elements of City.

Elements must be considered when determining relationships between attributes. The elements are not included in the data model because they are the physical data in the database. By knowing and understanding the elements of an attribute, you can better design your data model and project. As shown in the following data model, the attribute Division has multiple attribute elements, such as Men’s Clothing, Shoes, and Sporting Goods.
Another example displays the elements and data for the Store attribute. Each attribute element is a row in an attribute lookup table as shown in the following diagram:

Elements are typically referred to by their most descriptive form. For example, when you include “Atlanta” in a report, you are referring to the element that corresponds to the Store_Name “Atlanta.”
Attribute forms

Attribute forms are identifiers or descriptors of an attribute. Every attribute must have at least one form, and most have two:

- the ID form
- the primary description form

Every attribute must have an ID form, such as Customer_ID or City_ID. Some attributes can have additional descriptive forms that do not serve as the primary description form. For example, the Customer attribute in the MicroStrategy Tutorial has various forms. The forms include the ID and the Customer Name, which serves as the description of the attribute. Also included are address and e-mail.

One of the forms for the element City is Name. Chicago is the Name for the element of City with the ID of 1. Other forms for Chicago are a URL, such as www.chicago.com, or an Abbreviation such as CH.

Each attribute form provides details that identify and describe an attribute. The Store_ID is a unique numeric identifier for each store, while Store_Name holds the actual store name. Other attribute forms for the Store attribute can include ID, numbers, descriptive names, short abbreviated names, URLs, and so on. In MicroStrategy, you can assign a maximum of 32 forms per attribute.

This example uses the attribute Store and its corresponding elements and forms:
A simple lookup table with three columns holds the following separate forms:

<table>
<thead>
<tr>
<th>Store_ID</th>
<th>Store_Name</th>
<th>Store_Long_Name</th>
</tr>
</thead>
</table>

- Store_ID: a unique, identifying number for each store (*ID form*)
- Store_Name: the name of each store (*Description form*)
- Store_Long_Name: the full location, including the store name and state, of each store (*Long description form*)

In this example, the Lookup_Store table records all of the attribute form data for the Store attribute.

Attributes must contain at least one ID form, which uniquely identifies the attribute. The forms you create must have a reference to a lookup table and can include multiple expressions. Each table must have an ID form as that is how the table will be joined. You can choose a lookup table in the Attribute Editor from a list of tables existing in the project.

For example, two tables exist, one with the forms Customer_ID, Name, and SSN. The second lookup table contains Customer_ID and E-mail. The attribute will have four forms and the tables will join together through the ID columns.

**Attribute form properties**

When you create forms in the Attribute Editor, you must select properties for each form. These properties affect the display of the forms and include the following:

- Form categories help categorize the types of forms. The standard options are ID, Desc, and None. You can create new form categories in the Attribute Editor.
Format types control how the form is displayed. Format types also control how filters are built. For example, specifying a format type of Big Decimal allows users to preserve precision when qualifying on the form with more than 15 digits. Big Decimal is discussed in detail in Appendix B, Data types.

Default sort governs how the form is sorted by default when included in a report. You can choose from Ascending, Descending, or None.

### Attribute form expressions

Simply put, attributes act like holders of information and provide context for facts. For example, the customer attribute holds information about the customer such as Name and Address. These information units are called attribute forms. An attribute form expression defines what columns in the warehouse are used to represent the attribute form in SQL. Each attribute form must have at least one expression declared. Although you can have multiple expressions in different tables, a form cannot have two different expressions in the same source table.

You can create expressions using attribute columns, constants, and/or mathematical operators, for example, +, -, /, *. Only implicit attributes do not include a column in the expression, since they only use the constants you declare.

You can create a form expression using Apply functions. These functions are discussed in Appendix C, Pass-through Expressions.

The types of attribute form expressions are

- simple
- implicit
- derived
- heterogeneous mappings
Simple form expressions access data through columns you include when creating attributes. Implicit and derived attributes do not relate directly to data stored in the data warehouse. These attributes create virtual data by combining or using columns to generate the data.

**Simple expressions**

A simple expression is based on a single warehouse column. The definition of the simple expression includes the tables in which the column is found.

For example, Category is an attribute in the MicroStrategy Tutorial. It has two forms, ID and Description, both of which are defined by simple expressions. ID is based on the CATEGORY_ID column and Description on CATEGORY_DESC, both in the table LUCATEGORY.

**Implicit expressions**

An *implicit attribute* uses a virtual or constant attribute that does not physically exist in the database. Such an attribute has an implicit expression, which is a constant value, though nothing is saved in a column. For example, you can create “temporary columns” in the database with a value of “1” for every row, which simplifies COUNT limitations. So, in the Attribute Editor, you enter only a “1” in the expression to create a count.

Implicit attributes are useful in analyzing and retrieving information. When analyzing data, you can use constant attributes to create a COUNT to keep track of the number of rows returned. You can use constant attributes when building metrics, where you can sum the column holding the constant to create a COUNT. Any constant is acceptable, for example, RushOrder=’Yes’.
Derived expressions

A derived expression has its value determined by an expression which contains more than just a column in a table. Any operation on a column, such as adding a constant, adding another column, or setting the expression to be an absolute value, creates a derived expression. In other words, you are making a new expression from information that is already there.

For example, you can create a derived attribute to calculate age or anniversaries. By calculating the difference between the columns date of birth and current date, you can create an attribute to hold the Age that has been derived from the two columns. Calculations and functions used in the expression can assist in deriving data from the database by producing SQL with database-specific syntax.

As another example, the derived form expression “Name” consists of the two strings “First” and “Last”:

\[\text{Name} \rightarrow \text{First} + \ "\ + \text{Last}\]

On a report, this information is displayed as Mary Jones under the Name column.

Heterogeneous mappings

There are no restrictions on the names of the columns used in the expressions of a given attribute form. Heterogeneous mapping allows the Engine to perform joins on dissimilar column names. If you define more than one expression for a given form, heterogeneous mapping automatically occurs when tables and column names require it.

For example, because different source systems store Date information in various contexts, your company can have multiple columns in different tables that all represent the concept of Date. The ID form of the attribute Date may contain two expressions. The Day_Date column occurs in the LU_DATE table and the Order_Date column in the ORDER_DETAIL and ORDER_FACT tables.
Each expression is linked to a set of source tables that contain the columns used in the expression. Of all the tables in which the columns exist, you can select as many or as few as you want to be used as part of the attribute’s definition.

You can view the chosen tables in the source Tables area to the right of the Form Expressions area in the Attribute Editor.

The data types of columns used in a heterogeneous mapping for a given attribute must be identical or similar enough for your particular RDBMS to join them properly. For example, most databases cannot join a data type of Text to a data type of Number. However, depending on your database platform, you might be able to join between data types of Number and Integer.

Attributes and SQL

Reports are made possible by SQL. The user creates a report and then the Intelligence Server, using this report definition, instructs the engine how to build the SQL for that report. The expressions defined in an attribute or fact define the SELECT clause of a SQL command.

For example, consider the following:

```sql
Select Store_ID, Date, sum(Sales)
From Store_Fact
Group By Store_ID, Date
```

You have specified that you are looking for sales information by store and date. The attributes and metrics you have defined tell the Intelligence Server where to look in the data warehouse for the information and how to create the SQL that will retrieve it. Because of this process, you do not have to know SQL to extract information from your data warehouse.
Column alias

For attributes, a *column alias* performs the same function as it does for facts. By default, the data type for an attribute form is inherited from the data type of the column on which the form is defined. However, there are cases where you may need to change the data type. Following are some examples of such cases:

For example, in your warehouse you have a lookup table for Accounts where the ID is Account Number and the ID is stored in the database as DECIMAL(18, 0). Because this column stores high-precision values, you must modify the column alias for the attribute form and map it to a special data type, Big Decimal, so that precision can be preserved when performing filtering, drilling, or page by on the Account attribute.

Another example could be a case in which your warehouse does not have a lookup table for year information, but you would like to create a Year attribute. Many database platforms have functions that can extract parts of a date from a Date data type. For example, SQL Server has a Year function that extracts just the year from a date. In such a case, you can create a Year attribute using the following form expression:

```
ApplySimple("Year(#0)", [Date_Id])
```

The data type for this attribute is automatically set to a Date data type. This is because Date_ID is a Date data type. However, the result of the calculation is a year, such as 2002, and it is an integer.

When a temporary SQL table is created, if you do not change the data type of the column alias, the system uses a Date data type and tries to insert integer data into this column. While this does not create a problem in all database platforms, some databases will return an error. To avoid the possibility of an error due to conflicting data types, modify the column alias for the attribute form and change the default Date data type to an integer data type.
In addition to specifying the data type to be used for an attribute form, the column alias also lets you specify the actual column alias name to be used in the SQL generated by MicroStrategy. When you create a form expression using a custom expression as discussed above or using multiple columns, the column alias for the attribute form defaults to CustCol_1 (or CustCol_2, CustCol_3, and so on). The following piece of SQL shows, in bold, where the column alias name is used:

```
SELECT Year(a12.Date_Id) CustCol_1, 
    sum(a11.Tot_Dollar_Sales) WJXBFS1 
FROM YRCATEGORY SLS a11 
    cross join TRANS_DATE_LW_LY a12 
GROUP BY Year(a12.Date_Id)
```

While the column alias name does not affect the actual results or your report, you can change the column alias name to be more meaningful. The above example is a simple one, but this can be useful for troubleshooting the SQL for a particularly complex report.

**Form groups**

A *form group* is a grouping of attribute forms that have something in common. You can create form groups to combine forms that you want related. By grouping forms, you can design a uniquely defined form that groups two or more forms under an attribute. When you create a form group, the included forms are joined together and act as one. The forms in the form group can never be viewed separately once they become part of a group. See the following example of the Customer form group.
This form group joins the forms Last_Name and First_Name to identify the attribute Customer. To create a form group, choose the same form category for both forms. You are then prompted for a form group.

**Attribute relationships**

You link directly related attributes to each other by defining parent-child relationships. Attribute elements, or the actual data values for an attribute, dictate the relationships that you define between attributes. The parent-child relationships you create here determine the system hierarchy. In other words, they define how the engine generates SQL, how tables and columns are joined and used, and which tables are related to other tables.

**Joint child relationships**

Some attributes exist at the intersection of other indirectly related attributes. Such attributes are called *joint children*.

Joint child relationships are described in depth in Appendix D, *Advanced Data Modeling*. 
Attribute display

Once attributes are built, they are used in two primary ways—browsing and reporting. Each attribute can be displayed in a variety of forms so you must specify the default display of each of the attributes in the project. You can do this on a report-by-report basis, but you still must specify the global, or project-wide, default for each attribute.

You must choose a default attribute display for browsing and another for reporting. Report display forms are the attribute forms that appear as columns in a completed report. Browse forms are the attribute forms that appear as a user browses through the element list of an attribute in the Data Explorer. This separation allows for greater attribute display flexibility depending on the application.

The forms you select for an attribute determine which attribute elements are displayed when the report is executed. By selecting different forms for the attribute, you, in fact, select a different set of values for display. For example, in a report that includes Region as an attribute, if ID is selected as the attribute form, the display could be a number such as four. If Description is selected, the display could be a name, such as Northwest. If a report lists the cities in which you have stores, then you might choose to display the Long Description form, such as Chicago, instead of the URL attribute form, that is, www.chicago.com. When you modify the attribute display on a report, you can select, for each attribute, which attribute forms should appear on the report or you can use the attribute’s default setting.

You can also select which attribute forms are retrieved with the report results but not displayed on the grid, that is, they are found in Report Objects. In Grid view, you can add the attribute forms in Report Objects to the report without reexecuting the report.

You can modify the attribute form display by

- right-clicking an attribute on a report or template
- using the Attribute Display dialog box, accessed from the Data menu
For step-by-step instructions on displaying attribute forms on a report or template, see the online help.

**Using attribute forms versus characteristic attributes**

Attribute forms can be considered as additional descriptions for an attribute whereas characteristic attributes can be considered as report elements and group by elements that have a one-to-one or a one-to-many relationship with the attribute.

You should use an attribute form rather than a characteristic attribute in a report if:

- there is a one-to-one relationship between the attribute and the attribute form.
- you do not group by the attribute form.
- you do not need to change the attribute form on a report when you view the report on the Web. When you view a report on the Web, you have access only to the default reporting form of the attribute and cannot access the other characteristics embedded in the attributes.

**Compound attributes**

A *compound attribute* is defined as an attribute with more than one column specified as the ID column. This implies, of course, that more than one ID column is needed to uniquely identify the elements of that attribute. Generally, you build a compound attribute when your logical model reflects that a *compound key* relationship is present.
For example, a retail project has two attributes, Class and Item. Class is the parent of Item and has a one-to-many relationship with it. The values in the Item_ID column do not uniquely identify an item. The item shirt has an Item_ID of 1. However, there are different shirts, depending on the class—men’s, women’s, and children’s. Therefore, to uniquely identify a man’s shirt, Item_ID and Class_ID must be grouped together, creating a compound attribute.

All of the ID forms of the compound attribute should be grouped together. They should also use the same lookup table.

Example: creating a compound attribute

Report requirement

You need a report that contains sales figures by distribution center. Distribution center IDs are unique within each country, but the same values can exist in different countries.

How can you accomplish this?

Solution

Create distribution center as a compound attribute, with two attribute forms, ID and Description. When setting up the ID, select the source table columns for Country ID and Distribution Center ID. This creates a unique identifier for each distribution center, regardless of country.

Next, build a report using the distribution center attribute and a sales metric.
Introduction

An **HTML document** is an HTML container for formatting, displaying, and distributing multiple reports on the same page, or at the same time within a project. You can modify the appearance of an HTML document, just like any other HTML page, to include text, images, hyperlinks, tables, and one or more report objects.

The HTML document object, earlier called the document object, is the standard container for creating dashboards and scorecards to display a group of reports within the MicroStrategy platform. **Dashboards** or **scorecards** are popular means of displaying and distributing data from business intelligence projects. Scorecards typically follow a specific methodology and are focused on key metrics within a business area. Dashboards, on the other hand, tend to provide key metrics as well as summary information. While the business logic behind designing a dashboard or a
scorecard could be different, the technical implementation of the two is achieved in the same manner using MicroStrategy. Both dashboards and scorecards are a group of reports and metrics that are tied together by business logic. For the remainder of this chapter, the term dashboard is used to refer to such report groupings.

Typically, the end users for dashboards are high-level managers and executives. Requirements from such end users or business analysts are critical to the design of an effective dashboard.

This chapter first describes HTML document layout, creation, and viewing. It then discusses advanced XML and XSL concepts, which provide functionality for personalizing documents. Further, this chapter provides high level tips for designing and building a dashboard, along with detailed steps for implementing a gauge-based dashboard. Additionally, this chapter provides tips for simple XSL customization.

**HTML document layout**

The HTML document layout is used to position the reports inside the document. The layout is HTML, allowing you to insert images, text, tables, and hyperlinks; anything you can add to a Web page you can add to an HTML document.

The HTML document layout is an HTML file that includes special tags to identify the placement of the reports. Reports are represented by customized image tags. These images are replaced by the actual report when you execute the HTML document in HTML Document View in Desktop or through MicroStrategy Web.

HTML documents use XML and XSL to allow the user to view the content of the reports included in the HTML document. These technologies allow you to separate style from content when creating Web pages. XML defines the structure of information while XSL defines the formatting of information. The XML of the report is provided by the Intelligence Server, and the formatting is supplied by the XSL specified in the HTML Document Editor.
Advanced concepts: XML and XSL

You do not need to know anything about XML and XSL to successfully create and view HTML documents. However, the ability to customize the XSL provides additional functionality that you can use to create more personalized HTML documents.

XML

XML is an acronym for eXtensible Markup Language. XML provides a standard set of rules for representing data in a textual representation. Like a database table, XML contains both data and information about that data. For a database table, this information takes the form of column names and data types. For XML, it is stored as tags and attributes. A tag in XML is similar to a tag in HTML: it is not in itself data to be displayed or used, but rather provides information about how to display the data. An attribute in XML is similar to an attribute in HTML: it provides characteristics about a tag, and also about the underlying data. In XML, each piece of underlying data is called an element.

Attributes and elements in XML and HTML are not related to MicroStrategy attributes and elements.

XML can more easily represent a wider variety of data than a relational table can. This flexibility is one important part of what makes XML so powerful. The other part is the ability to make use of any custom tag within an XML document. Unlike HTML documents, which are limited to a predetermined set of tags, XML documents can include literally any tag within them; the interpretation of the tag is left to the XSL Stylesheet and the rendering application.
The XML generated for the document definition contains a pointer with a path to the HTML layout file. Therefore, the HTML file needs to be accessible from the MicroStrategy Intelligence Server and the MicroStrategy Desktop interface. This is also true for XSL files associated with the content elements. At run time, the MicroStrategy Intelligence Server scans through the HTML layout file and replaces the image placeholders with the corresponding reports and applies the given XSL to each of the reports.

For more information on the MicroStrategy XML tag definitions, please refer to the MicroStrategy SDK documentation.

There are also several publications available that provide additional information about the XML standard. In addition, the World Wide Web Consortium (W3C) publishes a set of Web pages at http://www.w3.org/XML/ documenting the standard and listing additional resources. Please refer to these outside sources for more information on XML.

**XSL**

XSL is an acronym for eXtensible Stylesheet Language. XSL is what dictates the style (such as color and font) for a grid. Each report in an HTML document must have an XSL associated with it so that the report can be formatted.

An XSL Stylesheet is a specific type of XML document and therefore must observe the same set of rules as any other XML document. The XSL standard provides a set of special tags, rules, and methods that can be used together to process XML documents and turn them into formatted output such as HTML.

For more information about the Extensible Stylesheet Language, please visit the W3C Web site at http://www.w3.org/Style/XSL/.
XSL stylesheets

XSL Stylesheets provide a very powerful means of controlling the output format for MicroStrategy grids. They can be used for much more than simple grid formatting control. For example, XSL Stylesheets can be used to control the insertion of images, phrases, or even frames.

For a list and description of the stylesheets that are installed with MicroStrategy in the XSL folder within the application directory (Drive:/Program Files/MicroStrategy/Desktop/XSLs, assuming you installed it in the default directory), refer to the MicroStrategy SDK documentation.

An HTML document must use the default stylesheet (default.xsl) for thresholds to be displayed as they are in Desktop or Web.

Creating HTML documents

You can create an HTML document to create a new dashboard. When you choose to create a new HTML document, the HTML Document Editor opens. The HTML Document Editor has two panes:

- HTML Document content pane allows you to see the properties of the reports contained in the HTML document. It becomes populated as you add reports to the document layout section.

- HTML Document layout pane is where you add objects to be included in the HTML document and adjust how they appear in the document. You can add reports, images, text, and tables to an HTML document, along with any other objects or information that can be included in an HTML page.

You can also display the Object Browser in the HTML Document Editor by choosing Object Browser Pane from the View menu.
HTML document views

You can work with HTML documents in the following views:

- Normal Edit View provides WYSIWYG (what you see is what you get) HTML editing. You can add tables, text, images, and reports. HTML is generated automatically. Note that when you drag and drop report objects while in this view, they are displayed with an icon placeholder. This placeholder is replaced with the report when you choose HTML Document View from the View menu or execute the HTML document in MicroStrategy Web.

- HTML Edit View displays the HTML source code for the HTML document. Edits made in the source code are immediately reflected in the Normal Edit View. The source HTML code can also be edited using third-party tools and then imported into the HTML Document Editor via the File menu.

- HTML Document View displays the HTML document and executes the reports. To print an HTML document showing report data instead of the placeholder icons, you must print in this view.

When you run an HTML document from Desktop, the report links and hyperlink drill paths appear to be enabled. However, you cannot view the report details or click the hyperlink drill paths to get the details of the report data at different levels. If you click on a report link or a hyperlink drill path, the browser page opens and displays "Page cannot be displayed". The report links and hyperlink drill paths work only when you view the document through MicroStrategy Web.

Report characteristics

When you select an object in the HTML Document Layout pane, you see the following report characteristics in the lower part of the document content section:

- Name is the name of the report.
- View allows you to view the report as a Grid or as a Graph in the document.

- Banding allows you to turn report banding on or off for the document. Banding changes the colors of rows in the document to visually separate them.

- XSL allows you to specify an XSL file to use to format the report in the HTML document.

You can modify any of the report characteristics by right-clicking the report in either the HTML Document Content pane or in the HTML Document Layout pane.

Additional information on reports and report characteristics may be found in Chapter 2, *Reports*.

### Image URLs

Since HTML documents can be viewed interactively using MicroStrategy Web or delivered to external sources using MicroStrategy Narrowcast Server, URLs and file paths must be constructed correctly. When HTML documents are viewed using MicroStrategy Web, the Web server that is transmitting the document can process relative URLs. This is not the case when HTML documents are delivered to users and viewed from external sources such as a mail server. These external sources can resolve only fully-qualified URLs, not relative URLs. A fully-qualified URL specifies the resource name plus all the subdirectories, the domain name, and HTTP or HTTPS, as in `http://www.microstrategy.com/images/image.gif`.

In contrast, a relative URL is a local URL from which certain information, such as directory names, is omitted. It is relative because the URL is only valid relative to the URL of the current resource. For example, `../image.gif` is located one level up from the current directory. If the current Web page is `http://www.microstrategy.com/images/intro.html`, then the file is saved in the `www.microstrategy.com/images` directory.
To avoid HTML documents that contain broken links, it is important to follow these rules:

- Use fully-qualified HTTP and HTTPS URLs to ensure accessibility for recipients outside of a company network.

- Only users who have network access to the shared image files will see images defined with fully-qualified file URLs using universal naming convention (UNC) file paths. An example of such an address is file://file_server/shared_directory/images/image.gif.

- Fully-qualified file URLs using shared drives work only for users who have the same shared drive defined. An example of an URL with a shared drive is file://x:\images\ images.gif.

- A relative URL must contain a base URL in the definition of the HTML page. If this base exists, the URL works correctly in Web pages that are sent via e-mail, performing like a fully-qualified URL as described above.

The HTML tag called base sets the base URL for relative URLs in an HTML page. An example of this tag follows.

```html
<HTML>
  <HEAD>
    <TITLE>HTML Page Title</TITLE>
    <BASE href="http://www.microstrategy.com/">
  </HEAD>
</HTML>
```
Best practices for creating dashboards

This section provides some tips for building dashboards.

Layout

You should keep the following in mind while preparing the layout of the HTML document:

- Organize data in tables.
- Provide a single page view and compact display.
- Choose a uniform coloring scheme.
- Use links wherever necessary.
- Use images and colors for better presentation and to provide visual cues.
- Use standard formats for information consumption.
- Highlight key information using thresholds.

To indicate thresholds you can use the images that are installed with MicroStrategy in the folder Drive:/Program Files/MicroStrategy/Analytics Modules/DocumentObjects/Images, assuming you installed it in the default directory. You must have the Analytics Modules installed to be able to use these images.

Parameters for dashboard design

This section describes the various parameters that you should consider while designing a dashboard. This is followed by an example from the Analytics Modules.
Target audience

You should first identify the target audience or end users who will consume the information in the dashboard. This could be employees at different levels within the organization, such as executives, managers, or certain groups within the company such as Marketing or Sales. In some cases, your organization may choose to provide this information to its partners or customers. The target audience dictates the type of data and the style of presentation.

Business purpose

In order to create an effective dashboard, you should identify the business purpose of the dashboard. For instance, your objective could be something broad-based such as getting an overview of the sales activity, or it could be something specific such as identifying those customers that pose collection problems.

Data set

Next, you should identify the data required to serve the identified business purpose. For instance, the information required may be sales data, customer lists, or web traffic data. The information should be available in a data warehouse for further utilization.

Analysis and workflow

You should then identify the set of analyses and build the required reports if the reports are not already available. You should also note any further analysis to be done using drilling, or links to additional details.
Graphics and numbers

For each of the reports or data points, you should identify the best mechanism for display. You can choose the appropriate graphs and images to highlight specific data, trends, and deviation from certain trends. Use grids in reports that require numeric details. Use thresholds within grids for highlighting deviations.

Delivery mechanism

You can then deploy the dashboard using Desktop, Web, or deliver it via e-mail using the Narrowcast Server.

Example from Analytics Modules

The following table shows the parameters for a dashboard from the Sales and Distribution Analysis Module and the Customer Analysis Module. Connect to MicroStrategy Analytics Modules for viewing these dashboards.

<table>
<thead>
<tr>
<th>Analytics Module</th>
<th>Sales Processing Scorecard</th>
<th>Customer Analysis Scorecard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target audience</td>
<td>Sales executives</td>
<td>Executives, managers in customer service and marketing</td>
</tr>
<tr>
<td>Business purpose</td>
<td>Get an overview of the entire sales process, measure total volume and efficiency</td>
<td>Identify customer churn and profitability</td>
</tr>
<tr>
<td>Data set</td>
<td>Quotation and order processing</td>
<td>Customer information and product sales</td>
</tr>
</tbody>
</table>
Dashboard reports typically have a target audience of executive level employees of an organization. Although you can deliver these reports in different formats, these reports are often noted for their simplicity to deliver powerful analytical information in a graphical summary for quick review.

To provide a graphical summary in the reports, you can set thresholds, using images in the format definition for the thresholds. If you accepted the default settings for installation, the images for the gauges can be found in C:\Program Files\MicroStrategy\Tutorial\images. You can copy these images to a machine on your network. A small sample of the available images is shown in the following figure.

<table>
<thead>
<tr>
<th>Analysis and workflow</th>
<th>Sales Processing Scorecard</th>
<th>Customer Analysis Scorecard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counts of inquiries, quotations, and orders</td>
<td>Revenue by region and month</td>
<td></td>
</tr>
<tr>
<td>Conversion rates for quotations and orders</td>
<td>Acquisition and attrition rates by Lifetime Value Score</td>
<td></td>
</tr>
<tr>
<td>Top 10 customers and products by profit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Graphics and Numbers | Use of images to illustrate the flow of the process and metric values to provide actual numbers | Graphs to highlight trends in customer churn and thresholds with gauges to show revenue trends |
To enhance the attractiveness of these reports, you can place the dashboard reports in an HTML document object, along with the images in an explanatory legend.

You can view a sample gauge-based dashboard, Electronics Dashboard, from the MicroStrategy Tutorial Project in the MicroStrategy Tutorial\Public Objects\Reports folder.

Example: implementing a gauge-based dashboard

Report requirement

The Senior Sales Executive of your company wants a report on the sell-through percentage of all the suppliers to view the sell-through percentage in broad ranges. You want to give the executive a graphical summary of the sell-through percentage.
How can you accomplish this?

**Solution**

Edit the sample report Supplier Sell-Through Percentage in the MicroStrategy Tutorial to add thresholds, or create your own report using thresholds.

If you run the sample report without any modifications and with the default filter, the result set is displayed in grid form as shown in the following figure.

<table>
<thead>
<tr>
<th>Supplier</th>
<th>Metrics</th>
<th>Sell-through Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bantam Books</td>
<td></td>
<td>31.8%</td>
</tr>
<tr>
<td>John Wiley &amp; Sons</td>
<td></td>
<td>10.6%</td>
</tr>
<tr>
<td>McGraw Hill</td>
<td></td>
<td>7.9%</td>
</tr>
<tr>
<td>Peripec</td>
<td></td>
<td>11.8%</td>
</tr>
<tr>
<td>Prentice Hall</td>
<td></td>
<td>14.5%</td>
</tr>
<tr>
<td>Scribner</td>
<td></td>
<td>18.6%</td>
</tr>
<tr>
<td>Simon &amp; Schuster</td>
<td></td>
<td>14.1%</td>
</tr>
<tr>
<td>Vintage Books</td>
<td></td>
<td>15.0%</td>
</tr>
<tr>
<td>Warner Books</td>
<td></td>
<td>9.6%</td>
</tr>
<tr>
<td>ACS Innovations</td>
<td></td>
<td>29.7%</td>
</tr>
<tr>
<td>ATE Electronics</td>
<td></td>
<td>44.3%</td>
</tr>
<tr>
<td>Audiotronics Inc.</td>
<td></td>
<td>33.3%</td>
</tr>
<tr>
<td>Digital Equipment</td>
<td></td>
<td>31.6%</td>
</tr>
<tr>
<td>Digital Office Inc.</td>
<td></td>
<td>34.9%</td>
</tr>
<tr>
<td>DSS Appliance Co.</td>
<td></td>
<td>31.2%</td>
</tr>
</tbody>
</table>

**To implement a gauge-based dashboard**

1. Right-click the Supplier Sell-Through Percentage report in the MicroStrategy Tutorial/Public Objects/Reports/Supplier Reports folder and select **Edit** to edit the report.

   You can also create your own report instead of using the sample report.

2. From the Grid menu, select **Thresholds**. The Thresholds dialog box opens.
3 Define the thresholds as desired.

4 In the Threshold list, select a threshold that you have defined.

5 From the Format Definition drop-down list, select **Image**.

6 From the Picture location drop-down list, select **Relative to HTML Document directory** as the method for retrieving the picture location.

   ![Threshold List](image)

   In addition to images on your own machine, you can choose images from a machine on the network or a website.

7 In the Source field, specify the location of the image.

8 Repeat steps 4-7 for each threshold you have defined.

9 Click **OK** and view the report in Grid view.
The report is displayed correctly only when you run it from MicroStrategy Web. In the MicroStrategy Desktop, the report appears as shown below.

<table>
<thead>
<tr>
<th>Supplier</th>
<th>Metrics</th>
<th>Sell-through Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bantam Books</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>John Wiley &amp; Sons</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>McGraw Hill</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>Perigee</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>Prentice Hall</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>Scribner</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>Simon &amp; Schuster</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>Vintage Books</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>Warner Books</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>ACS Innovations</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>ATF Electronics</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>Audiotronics Inc.</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>Digital Equipment</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>Digital Office Inc.</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
<tr>
<td>DSS Appliance Co.</td>
<td>&lt;IMAGE&gt;</td>
<td></td>
</tr>
</tbody>
</table>

You can now place this report in an HTML document. For more information, refer to *Example: building an HTML document*.

**XSL samples for simple customization**

This section covers sample XSL changes you can use to achieve simple customization for reports used in HTML document objects in order to build dashboards and scorecards.

To learn more about the report XML structure, refer to the *Software Development Kit for Web Developer Guide: Web Application Development* and the XML Structure Appendix of *MicroStrategy Web Customization Guide*. 
Changing color of report link

In the stylesheet default.xsl or simple.xsl, which is used to create a report link, search for the text `<xsl:attribute name="HREF">`. This is followed by the following tags among others.

```xml
<FONT FACE="Verdana,Arial,MS Sans Serif" SIZE="1" COLOR="#000000">
   <B><xsl:value-of
       select="/mi/rit/rd/mi/in/oi/@n" /></B>
</FONT>
```

You can suitably modify the FACE, SIZE, and COLOR to get the desired format for the report link.

Displaying report information

You can add the following statement to the XSL being used in order to display report description.

```xml
<xsl:template match="/">
   <xsl:value-of select="mi/in/oi/@des"/>
</xsl:template>
```

Substitute `@des` with `@n` to obtain report name for display.
Example: building an HTML document

Report requirement

You have two related reports, Sales by Season and Sales by Month, that you want to view at the same time. Although they are both grid reports by default, the information would be understood more quickly if they were displayed in graph format. Finally, your company uses a standard style for its Web pages. This style is saved in a file named CompanyStandard.HTML.

How can you accomplish this?

Solution

In the HTML Document Editor, create a new HTML document. Import the layout file, CompanyStandard.HTML, to set up your company’s standard style. Add any text you want. Add each report, changing the Desktop Object View to graph and applying XSL formatting for each.
Introduction

Hierarchies are groupings of attributes that can be displayed, either ordered or unordered, to reflect their relationships to other attributes. There are two types of hierarchies: user and system. A user hierarchy is unordered, and you can easily change its design to include additional attributes or limit user access. This type of hierarchy is created to provide flexibility in element browsing and report drilling. The system hierarchy is ordered and it is created automatically when you create new projects.

This chapter discusses types of hierarchies, displays, and how to browse in a hierarchy.
Types of hierarchies

There are two types of hierarchies:

- **System hierarchy**: The system hierarchy specifies an ordered set of all attributes in the project but does not define ordering or grouping among attributes. The system hierarchy represents the relationships as defined by the logical data model. There is only one system hierarchy in each project.

- **User hierarchy**: User hierarchies are named sets of attributes and their relationships, arranged in specific sequences for a logical business organization. They are user-defined and do not need to follow the logical model.

**System hierarchy**

The *system hierarchy* is the default hierarchy that MicroStrategy sets up for you each time you create a project. It contains all of the attributes in the project and is actually part of the definition of the schema. When you first create a project, it contains only the system hierarchy.

The system hierarchy holds information on the relationships between attributes in the project. The system hierarchy cannot be edited but is updated every time you add or remove children or parents in the Attribute Editor, or when you define children in the Project Creation Assistant.

The system hierarchy is useful in determining relationships between objects. Attributes from the system hierarchy do not need to be part of an explicitly-defined user hierarchy. Any attributes that are not assigned to a user hierarchy remain available to the system as report objects, filter conditions, and components of consolidations.

You can view the system hierarchy in the Data Explorer or in the Hierarchy Viewer, but not the Hierarchy Editor. The Hierarchy Viewer is accessed from Graphical View in the Schema menu.
**User hierarchies**

When you create user hierarchies, you can define the browse and drill relationships between attributes. Browsing occurs through the data explorer, whereas in drilling you are actually choosing to move to higher or lower levels on a report. You can create these hierarchies in the Hierarchy Editor using one or more attributes from the system hierarchy. All the attributes are listed for you to choose from.

A user hierarchy is the only type of hierarchy you can define, and you can create any number of user hierarchies for each project.

You should define user hierarchies that correspond to specific areas of your company business model and data warehouse schema.

**Hierarchy tools**

The following tools help you work with hierarchies:

- Data Explorer
- Hierarchy Viewer
- Hierarchy Editor (for user hierarchies only)

**Data Explorer**

The *Data Explorer* is a tool in the Object Browser that holds the system hierarchy and the user hierarchies. As a tool, it makes the hierarchies available for users to include in new reports. When you create a new project, the system hierarchy for that project is automatically placed in the Data Explorer. User hierarchies, however, are saved to the Hierarchies folder in the Object Browser. You can move user hierarchies to the Data Explorer folder, which is under the Hierarchies folder, in the Object Browser when you want them available for use in element browsing. Moving hierarchies to and from this folder allows you to keep some hierarchies visible to the user while hiding others.
Hierarchy Viewer

The **Hierarchy Viewer** graphically represents user hierarchies and the system hierarchy. In the system hierarchy, the connections between the attributes represent the parent-child relationships. In user hierarchies, the connections show the browse paths between the attributes. The Aerial perspective provides an overview of hierarchies; its decreased scale allows you to navigate through the entire project.

The Hierarchy Viewer is accessed from Graphical View in the Schema menu.

Hierarchy Editor

The **Hierarchy Editor** allows you to modify user hierarchies by adding and removing attributes. You can also perform the following actions to control hierarchy display:

- Lock a hierarchy.
- Limit a hierarchy.
- Filter a hierarchy.
- Set an entry point.

These properties are discussed in more detail in the *Hierarchy display* and *Entry point* sections following.

Hierarchy organization

The best design for a hierarchy is to organize or group attributes into logical business areas. For example, you can place related attributes into hierarchies by their level.
The example below demonstrates the Location and Customer hierarchies. Within the Location hierarchy, State, City, and Store are organized according to their relationships. The Customer hierarchy also groups together the attributes Company, Contact, and Customer.

Before MicroStrategy 7.0, hierarchies had to be separate and distinct from one another, follow the dimensional structure in the logical data modeling, and include at least one attribute. Beginning with MicroStrategy 7.0, however, they do not follow these rules. Hierarchies provide convenient navigation paths through the data.

## Hierarchy structure

The system hierarchy is a structure based on relationships you define between attributes. A user hierarchy allows you to logically define and order groups of attributes. Both the system and user hierarchies allow you to navigate and organize attributes in your project.

When you group attributes together into hierarchies, you are developing a working design of the display and browse functions of the attributes. In the example below, there are two instances of the Region hierarchy. One hierarchy demonstrates Region having multiple States and the States having multiple Stores.
This hierarchy allows you to create drilling and browsing options to the lower levels to view Region, State, and Store on a report. But if you only include Store in the Region hierarchy, as in the second example, then the only options for drilling or browsing are the Region and Store levels.

Hierarchy display

You can perform the following actions in the Hierarchy Editor to control hierarchy display:

- Lock a hierarchy.
- Limit a hierarchy.
- Filter a hierarchy.
- Set an entry point.

Locked hierarchy

A hierarchy is referred to as *locked* when at least one attribute within that hierarchy has the Element display option set to Locked. Locking a hierarchy prevents viewing elements of the specific attribute and any lower level attributes in the hierarchy. Anything higher in the hierarchy is still visible.
You can lock the hierarchy to restrict viewing elements for security reasons or to better manage lengthy hierarchies. By restricting the view of attribute elements in the Data Explorer, you can prevent the expansion of long attribute element lists that can consume system resources. When you set the element display to locked, a padlock icon displays next to the attribute name.

For example, the attribute Order is locked in the Data Explorer sample shown below. This may be to prevent unauthorized users from accessing sensitive information about customers.

```
Limited hierarchy

Another way to restrict the viewing of attribute elements in the Data Explorer is to limit the number of elements that display at one time. This method is useful when there are extensive attribute elements in a hierarchy. Instead of loading all attribute elements at once, you can set the limit to five or ten at time. You can then click the arrows to see the next set of attribute elements.
```
For example, the Chocolate subcategory contains many items. Rather than displaying all of them at once and overwhelming the user, a limit of five items has been set. The following graphic displays this view in the Data Explorer.

### Filtered hierarchy

You can add filters to a hierarchy to control how data is retrieved and displayed. With a filter you can choose exactly which attribute elements display. For example, you can filter a hierarchy so that data for only one quarter displays, or data for only a few individual days of one quarter. Filters make data retrieval faster by only allowing specific data to display. However, you cannot use a prompt-based filter to filter a hierarchy.

Each attribute in the hierarchy can have multiple filters applied to it. When filtering attributes in a hierarchy, you are limiting the elements of the data returned when you browse the Data Explorer. Whereas setting limits can reduce the number of elements displayed at one time, filters can limit the scope and return a subset of the results.

Filters increase efficiency when retrieving data. You can limit user access to parts of a hierarchy when you apply filters to attributes. The filters allow the Data Explorer to display only the criteria you select, and the user is unable to see additional data in the hierarchy.
For example, you want to view only those customers who are younger than 30 years old. First, create a filter on customer age less than 30. In the Hierarchy Editor, add the filter to the customer attribute. Update the project schema, and view the Customer hierarchy in the Data Explorer. Only those customers younger than 30 years old are displayed.

When adding filters to an attribute in a hierarchy, you need to make sure that each filter is relevant to the attribute’s information. MicroStrategy does not validate that the associated filter makes sense on that attribute. That is the responsibility of the user.

Entry point

An entry point is a shortcut to an attribute in the Data Explorer. Creating an entry point grants you faster access to the attribute without having to browse through multiple attributes to reach different levels of the hierarchy.

When you create a user hierarchy, the hierarchy, the attributes, and their elements display in the Data Explorer. When you set an attribute to be an entry point, you are creating a shorter route to access attributes. For example, a typical hierarchy is Time. When you click on Time, folders for each Year, such as 2003, 2002, and 2001, open. When you click on 2002, a folder for each Quarter, such as Q1, Q2, Q3, and Q4, opens. If you are seeking Week24, this means you need to open several levels of attributes to reach the correct data level, which is Week. If you set the attribute Week as an entry point, the folder Week displays in the Data Explorer at the same level as Year. If an attribute is not set to be an entry point, it displays in its normal hierarchy structure.

If you set a locked attribute as an entry point, it still displays in the hierarchy but a with padlock icon. You can see the locked entry point, but you are not able to access attributes below that level.
Hierarchy browsing

You can design the user hierarchy browsing for the Data Explorer by assigning browse attributes. A browse attribute is the attribute child defined for the hierarchy attribute. When you apply browse attributes to attributes in a hierarchy, you are specifying what levels of detail are visible when browsing the Data Explorer.

Once you choose which attributes to place in a hierarchy, you can define the relationships between them. These relationships determine how the users can browse the attributes from the Hierarchies folder. For example, if Catalog, Category, Subcategory, and Item are the attributes that comprise the user hierarchy Catalog Items, the hierarchy resembles the example below.

```
/   
|   
Catalog
    |
  Category
    |    |
Subcategory
    |    |
  Item
```

A user hierarchy does not need to have these relationships defined. It can simply be a collection of attributes.

For each attribute you select to be a part of the hierarchy, you can assign one or more browse attributes to it. For example, assume that the same attributes have been defined for the Catalog Items hierarchy. Some of these attributes have been assigned a browse attribute. For example:

<table>
<thead>
<tr>
<th>Hierarchy Attribute</th>
<th>Browse Attribute(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog</td>
<td>Category, Subcategory</td>
</tr>
<tr>
<td>Category</td>
<td>Subcategory</td>
</tr>
<tr>
<td>Subcategory</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td></td>
</tr>
</tbody>
</table>
The addition of these browse attributes allows you to see the Subcategory elements directly from the Catalog attribute, without having to first view the Category attributes. The new hierarchy can be represented as shown below.

<table>
<thead>
<tr>
<th>Hierarchy Attribute</th>
<th>Browse Attribute(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subcategory</td>
<td>Catalog, Item</td>
</tr>
<tr>
<td>Item</td>
<td></td>
</tr>
</tbody>
</table>

In the Data Explorer, it resembles the example below.

You can now view the subcategories in the Spring 97 catalog without first having to browse through the categories.
Drilling down using hierarchies

Drilling is a function in MicroStrategy reports that allows you to browse lower levels of attributes along predefined criteria. Depending on the level of the attributes included in the drilling specification, reports allow the user to drill down to lower levels of detail. Basically, when the user selects a drilling level, the reports refresh to display that level of detail.

To enable a user hierarchy as a drill path, you must select the user hierarchy to be used as a drill hierarchy in the Hierarchy Editor. Drilling is governed by the Enable Drilling privilege. If a user hierarchy is not selected, the default drill path is defined by the System Hierarchy. Also, the browsing attributes relationship in a user hierarchy can be viewed as a potential drilling path.

After a user hierarchy is enabled for drilling, it contributes to the drilling path of any attributes in it. For instance, assume Week is a browsing attribute assigned to Year. When a user right-clicks on Year and selects Drill Down, the attribute Week appears in the drill-down list.

Additional information on drilling is available in Chapter 14, *Drill Maps*. 
**Introduction**

Drill maps allow you to create fully customized drill paths that are available to your users while drilling on a report. By default, the paths available are based on the system hierarchy of the project. You can create custom drill maps that can override these defaults.

This chapter describes how a drill map works, how to create a drill map, and how it can affect what you see when drilling on a report.

**What is drilling?**

After executing a report in a MicroStrategy reporting environment, you may need to execute another report based on the original report to get more detailed or supplemental information. For example, after looking at annual sales of a certain city, you may want to look at the monthly sales for the same city. Alternatively, after noticing that a certain item had
a very high profit margin, you may want to see if that is also true for the entire category of that item. Such actions where you create a related report based on an existing report are referred to as *drilling*.

Even though a report generated as a result of drilling is related to the original report, they are, in essence, two entirely different reports. This means that the two reports can be saved or changed independent of each other. The two reports are different either because they have different templates or different filters, or both.

**Drill maps and drill paths**

*Drill maps* determine the options available to an end user when drilling on a report. By right-clicking on a report and choosing the drill option, you are using drill maps.

When the drill hierarchies are created, a default drill map is created. If no drill hierarchies are created, then the system hierarchy is used to create the default drill map. The drill map determines what options are available to you when you drill on a report object. These different options are referred to as *drill paths*, which includes the destination of the drill. The destination can be an attribute, a consolidation, a hierarchy, or a template.

In summary, a drill map determines what drill paths are available while drilling from a report object. By default, the drill paths available to a report object reflect exactly the drill hierarchies of the project.

**Default drill paths**

Before customizing your drilling options you need to understand how the default drill paths work.
The end user can drill from any object on a report, other than a simple metric. For example, drilling down from an attribute or hierarchy allows you to access other child attributes in the same hierarchy. Drilling from a consolidation allows access to the attributes that make up the consolidation. Note that by default in these types, drilling changes a report by navigating through the drill hierarchies and selecting another attribute to view. The original object is replaced with the one drilled to. Drilling on a compound metric allows you to view the metrics that compose it.

**Filters and drilling**

How a report’s filter is changed while drilling depends on what part of the original report is selected when the drill is performed. By default, if an attribute element on the original report is selected while drilling, then that attribute element is added to the new filter created for the drill. The filter from the original report on which you drill is carried over as well. For example, a report lists revenue by state and contains a filter for 2002. You select Virginia when you drill to store. The resulting report contains 2002 revenue for Virginia stores only. You can change this default behavior for a drill path in the Drill Map Editor and for a report in Report Data Options.

There are two ways to drill using the right-click menu. If you right-click a header, a filter is not added to the drill. If you right-click an attribute element, the filter is used.

**Creating custom drill maps and paths**

You can override the default drill map by creating your own custom drill maps and paths. Once you begin customizing a drill map for an object, none of the drill paths of the system hierarchy are available for drilling on that object. For example, before you create a drill map for the attribute Region, the default drill map is the system hierarchy, which allows drilling up to Country and down to Call Center. You create a drill map and add a drill path down to Employee. You cannot drill to Country or Call Center from Region unless you add them to your new drill map as well.
To create a custom drill path, you select a destination and drill path type and set properties.

**Destination**

The destination is the object which you will drill to in the report. This can be an attribute, consolidation, hierarchy, template, or another drill map. If the drill path is set to a template, every time you use this drill path a new report with the selected template is generated. When an existing drill map is chosen as the destination, it functions as a shortcut to the drill paths of the selected drill map.

For each drill path type, you can have multiple destinations. You can create multiple drill paths in each drill map.

**Drill path types**

A drill path can be one of the following types:

- **Up**—The destination can be any attribute or consolidation and does not have to be related to the original object. The destination is shown as part of the Drill Up menu when you right-click and select Drill in the report.

- **Down**—This is similar to Up, except that the destination is shown as part of the Drill Down menu when you right-click and select Drill.

- **Across**—This is also similar to Up, except that:
  - The destination is shown as part of the Other Directions menu when you right-click and select Drill.
  - A hierarchy can be used as a drill path.

- **Template**—This allows you to replace the template of the original report template with a completely different destination template. Select the template to use as the destination template.
• **Drill Map**—Use this as a shortcut to the drill paths of another drill map.
  
  – The destinations of those drill paths are displayed along with the destinations you have created. For example, you select a drill map that drills up to Brand. You already have a drill path up to Subcategory. When you select Drill and Up, both Brand and Subcategory are displayed.
  
  – Select an existing drill map to use as the destination.

You can group drill paths together in the right-click Drill menu by using the same Set Name for them. This is valid for all drill path types. Sets cannot cross drill types, so use them to group drill maps within a single drill type, such as Up.

**Drill path properties**

The following properties affect how the filter is manipulated:

• Apply user filtering conditions
• Apply original report filter conditions

These properties are not mutually exclusive; you have four combinations to choose from, which are listed below. The examples in the list are based on a report that lists revenue by state and contains a filter for 2002. Virginia is selected when the report is drilled to store.

• Apply both. This is the default. The resulting report contains 2002 revenue for Virginia stores only.

• Apply neither. The drill report includes revenue, by city, for all years and all states.

• Apply the user selection. The new report displays Virginia revenue for all years, listed by store.

• Apply the original only. The resulting report shows 2002 revenue by store for all states.
The other property that affects the filter is **Consider other filter qualifications when resolving metric qualifications in the new report**, which is related to the report filter’s advanced option. Both determine whether existing attribute qualifications are merged when the filter is evaluated. The report filter setting affects the entire report, while the Drill Map Editor setting applies only when you drill on the report. For more information on the report filter setting, see *Merge attribute qualifications* in Chapter 5, *Filters*. If you select Default in the Drill Map Editor, the report filter’s setting is used. Select Yes to consider other qualifications or No to ignore them, regardless of the report filter setting.

The Consider other filter qualifications property is applied only if Apply user filtering conditions, Apply original report filter conditions, or both these properties are selected.

For example, the following report contains a metric qualification for the top three revenue-producing regions. The metric qualification merges the qualifications when the filter is evaluated. This is the default setting.

![Customer Region]  
<table>
<thead>
<tr>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mid-Atlantic</td>
<td>$3,510,407</td>
</tr>
<tr>
<td>Central</td>
<td>$4,061,433</td>
</tr>
<tr>
<td>Southwest</td>
<td>$3,552,846</td>
</tr>
</tbody>
</table>

Drill down on the Central region to Customer City. The report shown below is displayed.

![Customer City]  
<table>
<thead>
<tr>
<th>Metrics</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>$290,346</td>
</tr>
<tr>
<td>Madison</td>
<td>$89,183</td>
</tr>
<tr>
<td>Minneapolis</td>
<td>$98,775</td>
</tr>
</tbody>
</table>

The top three revenue-producing cities in the Central region are selected and displayed. The qualifications were merged to produce this result, since by default the drill map uses the report filter’s merge attribute setting. In this case, it is the same as selecting Yes in the drill map setting.
Return to the original report, edit the drill map, and change the **Consider other filter qualifications** property to No. Again drill down on the Central region to Customer City. The following report is displayed:

![Customer City and Revenue table]

Only one city is displayed because the qualifications are not merged. First, the top three revenue-producing cities are identified, regardless of region. Then the drill to the Central region is applied to just those cities. Only one city, Chicago, of the three is in the Central region, so only that city is displayed on the final report.

The Keep parent object property determines whether the original object appears in the destination report. By default, this setting is not selected. To continue with the state revenue report example, the object name Virginia does not appear on the new report:

![State, Store, and Revenue table]

If the default setting is changed, Virginia does appear on the report:

![State, Store, and Revenue table]

This setting does not apply to the template drill type.
The Keep thresholds when drilling property retains thresholds during a drill.

The Priority setting affects how the drill path is displayed in a report:

- **Low**: The drill path is available as a right-click menu option in a MicroStrategy Desktop report. In a MicroStrategy Web report, this drill path is not available as a right-click menu option but can be accessed from the More Options link.
- **Medium**: The drill path is available as a right-click menu option in both Desktop and Web reports.
- **High**: The drill path is used as the default drill path in both Desktop and Web reports. It is still available as a right-click menu option.

When a user double-clicks an object on a Desktop report, the default drill path is used. In Web, if an object on a grid has a default drilling option, the elements of that object appear as hyperlinks on the grid. Click the hyperlink to drill on the elements.

To set a drill path as the default, assign its priority to High. Only one high priority drill path can exist in each drill map.

**Drill map association**

The drill map *association* defines which grid unit uses this drill map. In other words, drilling uses the object’s associated drill map. An object can have an association both on the object level and on each template/report. If there is no association on the template/report level, then the association on the object level is used when a user drills at that object.

If an object is already associated with a drill map, that drill map is displayed in the Drill Map Editor. Otherwise, the default drill map is based on the system hierarchy. Once you begin to modify the default, it is no longer the system hierarchy—the name changes automatically, although you can edit this preset name. When you save your changes, a new drill map is created.
You can create and edit a drill map as a stand-alone object from the Desktop. You can associate multiple objects with the same drill map, using the Associate with button. The associated objects appear in the Origin list on the interface.

You can also access the Drill Map Editor from the:

- Attribute Editor
- Consolidation Editor
- Report Editor
- Template Editor

When you edit a drill map from another editor, the drill map is associated with the object selected in the other editor. You cannot change the drill map association (that is, what other objects are associated with this drill map), but you can change which drill maps are associated with the selected object. For example, if you are editing the Store State attribute and you access the Drill Map Editor, only Store State is associated with the drill map you create.

If the original object is a template or report, the children objects are also available. For example, a sales report contains Store, Store State, Item, and the Revenue metric. You can create a drill map for each of the grid units except Revenue, because you cannot create drill maps for metrics.

Levels of drill map association

When you change or customize the drill map associated with a grid unit, you can do so at several different levels:

- **Project level**—If a drill map is associated with grid units at the project level, then all of the grid units in the project will have this drill map. Therefore, when you drill on a report, the default drill paths are those specified in this drill map. This option is found in Project Configuration.
• **Grid unit level**—A drill map can be associated with individual grid units such as attributes, consolidations, and hierarchies. When the object is used in a report or template, the grid unit level drill map overrides the project level drill map.

• **Template level**—If a drill map is associated with grid units on a particular template, it overrides the project level and grid unit level drill maps. The drill paths of this drill map are available in all reports that use this template.

• **Report level**—If a drill map is associated with grid units on a report level, it overrides the drill maps defined at the project level, grid unit level, and the template level. If a grid unit is not associated with a drill map at the report level, it inherits the map from the report template. If it is not associated with a drill map through the template, then the grid unit drill map is used, and so on.

The Drill Map Editor represents these levels and inheritances when you edit a report’s drill maps. If the Name field is greyed out (disabled), the selected report object inherited that drill map from the project, grid unit, or template. When you overwrite it by adding a different drill map to the object, the Name field is enabled.

For example, you create a drill map named Customer Attribute Drills for the attribute Customer. Create a report named Customer Revenue that displays Region, Customer, and the Revenue metric. When you edit the drill maps for the report and select Customer, the Name field is disabled but displays Customer Attribute Drills. No drill paths are displayed for the drill map. Because this attribute does not have a drill map on the report, it inherited the drill map from the attribute level. Because you cannot edit the attribute’s drill map from the attribute on the report, the Name field is disabled and the drill paths do not appear. When a new drill map is created for Customer on this particular report, Name is enabled, defaulting to Customer Sales Customer Drill Map.

By default, there is a project drill map containing all of the hierarchies that have been specified as drill hierarchies using the Hierarchy Editor in the project. It cannot be deleted, but it can be modified and overridden.
Removing associations

The **Remove Current Association** option disassociates the object from the current drill map and replaces it with its default map. Depending on the levels described above, this default map could be the template drill map, the grid unit drill map, or the project drill map.

The **Clear All** option deletes all the drill path information for the whole drill map. The object effectively has no drilling options. **Reset** reverses any changes and resets the drill map to the condition of the last save. Drill map associations are reset as well.
Logical Tables

Introduction

Logical tables represent tables in the data warehouse. There are three types of logical tables in the MicroStrategy environment: logical tables, table aliases, and logical views. While logical tables are set up in a project by using the Warehouse Catalog, logical views are created using the Table Editor. Different from the logical tables, which point to physical tables in the data warehouse, logical views are defined using the SQL queries against the data warehouse.

This chapter is intended to introduce the different types of logical tables, with a focus on how you can use the logical view feature to take advantage of the enhanced schema support by MicroStrategy.
Logical tables

Logical tables are MicroStrategy objects that form the foundation of a schema. While physical tables in a data warehouse consist of columns, logical tables in the MicroStrategy schema consist of attributes and facts. These attributes and facts are part of the report definition that the MicroStrategy Engine refers to when a report is executed.

There are three types of logical tables:

1. **Logical table**: is created for each physical table that is imported into a project, using the Warehouse Catalog. This type of logical tables maps directly to physical tables in the data warehouse. These physical tables are referenced in the SQL that is generated for the report.

2. **Table alias**: is created outside of the Warehouse Catalog and points directly to a physical table. A table alias can have a different name from the physical table. One physical table can have more than one table aliases. Table aliasing is used to create attribute roles (see Appendix D, *Advanced Data Modeling*).

3. **Logical view**: does not point directly to a physical table and is defined using a SQL query against the warehouse. Once created, the logical view can be used in the same way as the Type 1 logical table, based on which attributes, facts, and other schema objects can be defined. The logical view is also referenced in the SQL that is generated for the report; the whole SQL query is displayed in the place of physical tables as for Type 1 logical tables. Logical views are created using the Table Editor.

Using the Table Editor, you can view the content of all the logical tables as well as their associated warehouse tables. In the MicroStrategy Tutorial, logical tables and all the other schema objects are stored in the Schema Objects folder.
How should I use logical tables?

The most common logical tables are the ones that are imported into the project from the data warehouse using the Warehouse Catalog (accessed from the Schema menu). Based on these tables, you can create MicroStrategy schema objects, such as attributes and facts. For more information on how to use the Warehouse Catalog, please refer to the MicroStrategy online help (search for “Warehouse Catalog”).

When an attribute plays more than one role, you need to create an attribute in the logical model for each of the roles. One way to do it is to create explicit table aliases. Basically, you create multiple logical tables pointing to the same physical table and define those logical tables as the lookup tables for the attributes in different roles. For example, if the Customer table is used to represent both Ship to Customer and Bill to Customer, you can create a table alias to resolve the double usage case. First, create a table alias by copying an existing logical table and giving it a new or different name; then define the new attributes using the appropriate tables. For detailed information on Attribute Roles, please refer to Appendix D, Advanced Data Modeling, in this guide. To create a table alias, right-click the logical table name and select Create Table Alias. For step-by-step instructions, please refer to the online help.

Logical views are a little different from the above-mentioned logical tables and table aliases for the following reasons:

- Logical views do not map directly to physical tables in the data warehouse.
- Logical views are defined using SQL queries.
- Logical views are created from scratch, instead of being imported from a data warehouse or duplicated from existing logical tables.

However, once logical views are created, they can be used in the same way as the other logical tables, which means that you can use them to build attributes and facts and that you can also create table aliases for them.
The biggest benefit of using logical views is that you can model a MicroStrategy schema that cannot be supported with only the physical database structures in the warehouse. There are many common modeling scenarios that are easier to manage with the use of logical views, such as the following:

- slowly-changing dimensions
- attribute form expressions from multiple tables
- consolidated dimension tables
- recursive hierarchies

For common usage examples, please refer to the Logical view examples subsection in this chapter.

Whenever you create or add logical tables, table aliases, or logical views to the project, you need to update the schema. The Update Schema option can be accessed from the Schema menu.

Creating logical tables

As mentioned previously, most logical tables are brought into the project by using the Warehouse Catalog, and table aliases are created by duplicating existing logical tables. Detailed instructions on how to create them are provided in the online help (search for “Tables”).

Logical views, on the other hand, are created on Desktop using the Table Editor. One way to access the Table Editor is to select New from the File menu and choose Logical Table. The creation process involves a few simple steps that require you to provide your own SQL statement and map the columns in the statement to the correct data types.

When mapping the columns, if you use a column from an existing table in the Warehouse Catalog, you inherit the data type of that column. Keep in mind that if you change the data type, this change will affect all the tables with this column.

For step-by-step instructions, please refer to the online help (search for “Creating logical views”).
Using SQL for logical views

Since SQL queries are the key to creating logical views, you should be experienced with using SQL. It is your responsibility to ensure the accuracy and validity of your SQL statements. In addition, you should also understand that the SQL query entered for logical views is not modified in any way by MicroStrategy. Therefore, make sure that your RDBMS is optimized to answer the query that you create.

Because the MicroStrategy Engine does not parse through the SQL syntax, therefore, the statistics log does not contain any information about the actual physical tables accessed; the logical view is logged instead. The same would be true if you used a view in the database, in which case table objects accessed would not be logged either.

In the SQL generated for a report, logical views are generated as either a derived table or a common table expression (CTE) depending on the type of database that you use. It is recommended that you use derived tables to define logical views, although CTEs are also supported by some databases. Please check your database for best usage.

Logical view examples

The following business cases are intended to help you understand how you can use the logical view feature in your applications.

Business case 1: Distinct attribute lookup table

Many star schemas feature a single lookup table that is shared by all the attributes in one dimension (see the following example). While it is possible to model a schema with such a dimension table, often two problems arise:

- The model cannot support fact tables at the level of attributes that are not keys. This restriction applies to summary tables as well as to intermediate results that may be generated by the SQL Engine.
Usually, in one-SQL-pass reports, the MicroStrategy Engine joins the fact table with one lookup table and does the aggregation. If there is no distinct list of attribute elements, you may double count if you have to join to a table where that attribute is part of the key.

- Too many rows in the dimension table may slow down the SELECT DISTINCT query, thus affecting element browsing requests that display a list of attribute elements, for example, when populating pick lists for prompts.

The following is an example lookup table for Store, Market, and Region.

**Lookup_store**

<table>
<thead>
<tr>
<th>Store_ID</th>
<th>Store_Name</th>
<th>Market_ID</th>
<th>Market_Name</th>
<th>Region_ID</th>
<th>Region_Name</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this table, Market and Region are not the keys. Therefore, if the requested fact table is at the Market or Region level, a direct join between the fact table and the above lookup table may result in double-counting. To avoid that, you can use the Logical View feature to define another logical table Lookup_Market as follows:

```sql
Select Market_ID, Market_Name, Region_ID
From Lookup_store
Where level=1
```

Then use this table as the lookup table for Market. When it is joined with a Market-level fact table (Market_Sales), the following report SQL is generated:

```sql
Select a11.Market_ID, a11.Market_Desc,
    SUM(a12.Sales)
From (select Market_ID, Market_Name, Region_ID
    from Lookup_Store
    where level=1) a11,
    Market_Sales a12
Group by a11.Market_ID,
    a11.Market_Name
```
Business case 2: Attribute form expression across multiple tables

Attribute form expression across multiple tables is a very common request among customers. Usually, the case is on Date columns. For example, you want to define an attribute based on the Date difference between two Date columns (Ship_Date and Order_Date) in two different tables as follows.

F_Table1

<table>
<thead>
<tr>
<th>Ship_Date</th>
<th>Order_ID</th>
<th>Fact1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F_Table2

<table>
<thead>
<tr>
<th>Order_Date</th>
<th>Order_ID</th>
<th>Fact2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using the Logical View feature, you can use the following SQL query to create a logical table to calculate the Date difference and then define the attribute on that new column:

```
Select Ship_Date-Order_Date Cycle_time,
     F_table1.Order_ID, Fact1, Fact2
From F_table1, F_table2
Where F_table1.Order_ID=F_table2.Order_ID
```

The new logical table (logical view) looks like the following table, and a new attribute can be defined on the Cycle_Time column.

Logical view

<table>
<thead>
<tr>
<th>Cycle_Time</th>
<th>Order_ID</th>
<th>Fact1</th>
<th>Fact2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Business case 3: Slowly changing dimensions

Slowly changing dimensions (SCDs) are a common characteristic in many business intelligence environments. Usually, dimensional hierarchies are presented as independent of time. For example, a company may annually reorganize their sales organization or recast their product hierarchy for each retail season. “Slowly” typically means after several months or even years. Indeed, if dimensional relationships change more frequently, it may be better to model separate dimensions.

SCDs are well documented in the data warehousing literature. Ralph Kimball has been particularly influential in describing dimensional modeling techniques for SCDs (see The Data Warehouse Toolkit, for instance). Kimball has further coined different distinctions among ways to handle SCDs in a dimensional model. For example, a Type I SCD presents only the current view of a dimensional relationship, a Type II SCD preserves the history of a dimensional relationship, and so forth.

The discussion below is based on an example sales organization that changes slowly in time as the territories are reorganized, for example, sales representatives switch districts in time.

As-is vs. as-was analysis

One of the capabilities available with slowly changing dimensions is the ability to perform either “as-is” analysis or “as-was” analysis.

- “As-is” analysis presents a current view of the slowly changing relationships. For example, show me sales by District according to the way Districts are organized today.

- “As-was” analysis presents a historical view of the slowly changing relationships. For example, show me sales by District according to the way Districts were organized at the time the sales transactions occurred.
The techniques described here provide the flexibility to perform either type of analysis. They also provide you an easy way to specify which type of analysis you would like to perform.

**Example 1: Compound key with Effective Date and End Date**

One way to physically store an SCD is to employ Effective Date and End Date columns that capture the period of time during which each element relationship existed. In the example below, Sales Rep Jones moved from District 37 to District 39 on 1/1/2004, and Kelly moved from District 38 to 39 on 7/1/2004.

For information on compound keys, please refer to the Attributes chapter and Appendix D, *Advanced Data Modeling*, in this guide.

**LU_SALES_REP**

<table>
<thead>
<tr>
<th>Sales_Rep_ID</th>
<th>Sales_Rep_Name</th>
<th>District_ID</th>
<th>Eff_Dt</th>
<th>End_Dt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jones</td>
<td>37</td>
<td>1/1/1900</td>
<td>12/31/2003</td>
</tr>
<tr>
<td>2</td>
<td>Smith</td>
<td>37</td>
<td>1/1/1900</td>
<td>12/31/2099</td>
</tr>
<tr>
<td>3</td>
<td>Kelly</td>
<td>38</td>
<td>1/1/1900</td>
<td>6/30/2004</td>
</tr>
<tr>
<td>4</td>
<td>Madison</td>
<td>38</td>
<td>1/1/1900</td>
<td>12/31/2099</td>
</tr>
<tr>
<td>1</td>
<td>Jones</td>
<td>39</td>
<td>1/1/2004</td>
<td>12/31/2099</td>
</tr>
<tr>
<td>3</td>
<td>Kelly</td>
<td>39</td>
<td>7/1/2004</td>
<td>12/31/2099</td>
</tr>
</tbody>
</table>

When using this type of dimensional lookup table, the fact table must include a date field, such as a transaction date.

**FACT_TABLE**

<table>
<thead>
<tr>
<th>Sales_Rep_ID</th>
<th>Trans_Dt</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9/1/2003</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>9/10/2003</td>
<td>200</td>
</tr>
</tbody>
</table>
To specify the MicroStrategy schema

1. Create a logical view to represent just the current District-Sales Rep relationships.

   **LVW_CURRENT_ORG**
   
   ```sql
   select Sales_Rep_ID, District_ID
   from LU_SALES_REP
   where End_Dt = '12/31/2099'
   ```

2. Create another logical view that performs the “as-was” join between the lookup table and fact table, resulting in a fact view at the District level.

   The resulting view is an “as-was” or historical view, which captures the Sales Rep-District relationships that existed at the time the transactions occurred.

   **LVW_HIST_DISTRICT_SALES**
   
   ```sql
   select District_ID, Trans_Dt, sum(sales)
   from LU_SALES_REP L
   join FACT_TABLE F
   on (L.Sales_Rep_ID = F.Sales_Rep_ID)
   where F.Trans_Dt between L.Eff_Dt and L.End_Dt
   group by District_ID, Trans_Dt
   ```
3 Create a table alias LU_CURRENT_DISTRICT for LU_DISTRICT.

4 Define the following attributes:
   - Sales Rep
     - @ID = sales_rep_id; @Desc = sales_rep_name
     - Tables: LU_SALES_REP (lookup), LVW_CURRENT_ORG, FACT_TABLE
   - Current District
     - @ID = district_id; @Desc = district_name
     - Tables: LU_CURRENT_DISTRICT (lookup), LVW_CURRENT_ORG
     - Child: Sales Rep
   - Historical District
     - @ID = district_id; @Desc = district_name
     - Tables: LU_DISTRICT (lookup), LU_SALES_REP, LVW_HIST_DISTRICT_SALES
     - Child: Sales Rep
   - Date
     - @ID = date_id, trans_dt
     - Tables: LU_TIME (lookup), FACT_TABLE, LVW_HIST_DISTRICT_SALES
   - Month
     - @ID = MONTH_ID
     - Tables: LU_TIME (lookup)

5 Define the Sales fact:
   - Expression: sales
   - Tables: FACT_TABLE, LVW_HIST_DISTRICT_SALES
6 Define the metric as required:

- **Sales**: SUM(sales)

The result of this is a logical schema that looks like the following:

<table>
<thead>
<tr>
<th>LU_CURRENT_DISTRICT</th>
<th>LU_CURRENT_ORG</th>
<th>LU_SALES_REP</th>
<th>FACT_TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current District</td>
<td>Sales Rep</td>
<td>Historical District</td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Current District</td>
<td></td>
<td>Sales</td>
</tr>
</tbody>
</table>

**As-was analysis**

Specify the “as-was” analysis by using the Historical District attribute on reports.

- **Report definition**: Historical District, Month, Sales
- **Resulting SQL**

```sql
Select a11.District_ID District_ID, 
    max(a13.District_Name) District_Name, 
    a12.Month_ID Month_ID, 
    sum(a11.SALES) WJXBFS1
From (select District_ID, Trans_dt,sum(sales) sales 
    from LU_SALES_REP L 
    join FACT_TABLE F 
    on (L.Sales_rep_ID = F.Sales_rep_ID) 
    where F.trans_dt between L.EFF_DT and L.END_DT 
    group by District_ID, Trans_dt) a11 
```
join LU_TIME a12  
on (a11.Trans_dt = a12.Date_ID)  
join LU_DISTRICT a13  
on (a11.District_ID = a13.District_ID)  
group by a11.District_ID,  
a12.Month_ID

• **Report results**

<table>
<thead>
<tr>
<th>Historical District</th>
<th>Metrics</th>
<th>200309</th>
<th>Sales 200403</th>
<th>200409</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td></td>
<td>300</td>
<td>250</td>
<td>125</td>
</tr>
<tr>
<td>Southeast</td>
<td></td>
<td>150</td>
<td>300</td>
<td>150</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td></td>
<td></td>
<td>200</td>
<td>275</td>
</tr>
</tbody>
</table>

**As-is analysis**

Specify the “as-is” analysis by using the Current District attribute on reports.

• **Report definition**: Current District, Month, Sales

• **Resulting SQL**

select a12.District_ID District_ID,  
max (a14.District_Name) District_Name,  
a13.Month_ID Month_ID,  
sum(a11.SALES) WJXBFS1  
from  FACT_TABLE a11  
join (select Sales_rep_ID, District_ID  
from LU_SALES_REP  
where END_DT = '12/31/2099')a12  
on (a11.Sales_Rep_ID =  
a12.Sales_Rep_ID)  
join LU_TIME a13  
on (a11.Trans_dt = a13.Date_ID)  
join LU_DISTRICT a14  
on (a12.District_ID = a14.District_ID)  
group by a12.District_ID,  
a13.Month_ID
**Example 2: New surrogate key for each changing element**

A more flexible way to physically store a SCD is to employ surrogate keys and introduce new rows in the dimension table whenever a dimensional relationship changes. Another common characteristic is to include an indicator field that identifies the current relationship records. An example set of records is shown below.

**LU_SALES_REP**

<table>
<thead>
<tr>
<th>Sales_Rep_CD</th>
<th>Sales_Rep_ID</th>
<th>Sales_Rep_Name</th>
<th>District_ID</th>
<th>Current_Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Jones</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Smith</td>
<td>37</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Kelly</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Madison</td>
<td>38</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Jones</td>
<td>39</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>Kelly</td>
<td>39</td>
<td>1</td>
</tr>
</tbody>
</table>

When using this type of dimensional lookup table, the fact table must also include the surrogate key. A transaction date field may or may not exist.
FACT_TABLE

<table>
<thead>
<tr>
<th>Sale-Rep_CD</th>
<th>Sale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>150</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
</tr>
<tr>
<td>3</td>
<td>300</td>
</tr>
<tr>
<td>2</td>
<td>125</td>
</tr>
<tr>
<td>6</td>
<td>275</td>
</tr>
<tr>
<td>4</td>
<td>150</td>
</tr>
</tbody>
</table>

**Specifying the MicroStrategy schema**

1. Create a logical view to represent just the current District-Sales Rep relationship.

   **LVW_CURRENT_ORG**

   ```sql
   select Sales_rep_ID, District_ID
   from LU_SALES_REP
   where Current_flag = 1
   ```

2. Create a table alias LU_CURRENT_DISTRICT for LU_DISTRICT.
3 Define the following attributes:

- **Sales Rep Surrogate**
  - @ID = sales_rep_cd
  - Tables: LU_SALES_REP (lookup), FACT_TABLE

- **Sales Rep**
  - @ID = sales_rep_id; @Desc = sales_rep_name
  - Tables: LU_SALES_REP (lookup), LVW_CURRENT_ORG
  - Child: Sales Rep Surrogate

- **Current District**
  - @ID = district_id; @Desc = district_name
  - Tables: LU_CURRENT_DISTRICT (lookup), LVW_CURRENT_ORG
  - Child: Sales Rep

- **Historical District**
  - @ID = district_id; @Desc = district_name
  - Tables: LU_DISTRICT (lookup), LU_SALES_REP
  - Child: Sales Rep

- **Date**
  - @ID = date_id, trans_dt
  - Tables: LU_TIME (lookup), FACT_TABLE

- **Month**
  - @ID = MONTH_ID
  - Tables: LU_TIME (lookup)
  - Child: Date
4 Define the Sales fact:

- **Expression**: sales
- **Tables**: FACT_TABLE, LVW_HIST_DISTRICT_SALES

5 Define the metric as required:

- **Sales**: SUM(sales)

The result is a logical schema as follows:

As-was analysis

Specify the “as-was” analysis by using the Historical District attribute on reports.

- **Report definition**: Historical District, Month, Sales

- **Resulting SQL**

```sql
select a12.District_ID District_ID,
       max(a14.District_Name) District_Name,
       a13.Month_ID Month_ID,
       sum(a11.SALES) WJXBFS1
from FACT_TABLE a11
     join LU_SALES_REP a12
       on (a11.Sales_Rep_CD = a12.Sales_Rep_CD)
     join LU_TIME a13
```
on (a11.Trans_dt = a13.Date_ID)
join LU_DISTRICT a14
on (a12.District_ID =
a14.District_ID)
group by a12.District_ID,
a13.Month_ID

- **Report results**

<table>
<thead>
<tr>
<th>Historical District</th>
<th>Metrics Month</th>
<th>200309</th>
<th>200403</th>
<th>200409</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northeast</td>
<td></td>
<td>300</td>
<td>250</td>
<td>125</td>
</tr>
<tr>
<td>Southeast</td>
<td></td>
<td>150</td>
<td>300</td>
<td>150</td>
</tr>
<tr>
<td>Mid-Atlantic</td>
<td></td>
<td></td>
<td>200</td>
<td>275</td>
</tr>
</tbody>
</table>

**As-is analysis**

Specify the “as-is” analysis by using the Current District attribute on reports.

- **Report definition**: Current District, Month, Sales

- **Resulting SQL**:

```sql
select a13.District_ID District_ID,
       max(a15.District_Name) District_Name,
       a14.Month_ID Month_ID,
       sum(a11.SALES) WJXBFS1
from FACT_TABLE a11
join LU_SALES_REP a12
    on (a11.Sales_Rep_CD =
a12.Sales_Rep_CD)
join (select Sales_rep_ID, District_ID
      from LU_SALES_REP
      where current_flag = 1)
    a13
    on (a12.Sales_Rep_ID =
a13.Sales_Rep_ID)
join LU_TIME a14
    on (a11.Trans_dt = a14.Date_ID)
join LU_DISTRICT a15
    on (a13.District_ID =
a15.District_ID)
group by a13.District_ID,
a14.Month_ID
```
### Business case 4: One-to-many transformation tables

In order to support time-series analysis, such as month-to-date and year-to-date calculations, you need to define transformations. Although one-to-one transformations, such as Last Month, can be defined in terms of an expression, one-to-many transformations require tables in the database that map each date to all the previous dates that make up “month-to-date”.

If you do not already have such a table in the warehouse and your circumstances do not allow you to add additional tables to the database, then you can use the logical view approach to address this issue as long as you already have a lookup table for the Date attribute.

The SQL below can be used to define a logical MTD_DATE table, which contains the Date attribute. The MTD transformation can then be defined using the MTD_DATE column.

```sql
Select day_date day_date, B.day_date mtd_date
From lu_day A, lu_day B
Where A.day_date >= B.day_date
   And MONTH(A.day_date) = MONTH(B.day_date)
```

The same technique can be used to define a year-to-date transformation.

```sql
Select A.day_date day_date, B.day_date ytd_date
From lu_day A, lu_day B
Where A.day_date >= B.day_date
   And YEAR(A.day_date) = YEAR(B.day_date)
```
**Business case 5: Outer joins between attribute lookup tables**

A common request is the ability to generate an outer join between attribute lookup tables for a report that contains only attributes (that is, no metrics). For example, consider the tables below.

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>EMERGENCY CONTACT</th>
<th>DEPARTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP_ID</td>
<td>EMP_ID</td>
<td>DEPT_ID</td>
</tr>
<tr>
<td>FIRST_NAME</td>
<td>CONTACT_FIRST_NAME</td>
<td>DEPT_NAME</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>CONTACT_LAST_NAME</td>
<td>BUS_UNIT_ID</td>
</tr>
<tr>
<td>HIRE_DATE</td>
<td>CONTACT_PHONE_NUMBER</td>
<td></td>
</tr>
<tr>
<td>DEPT_ID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Given this structure, you could model an attribute hierarchy as follows:

- Business Unit -< Department -< Employee
- Hire Date -< Employee
- Emergency Contact -< Employee

In addition, the relationship between Employees and Emergency Contacts is such that each employee may have up to one contact, which means not all employees have contacts on record. One of the reports you probably would like to create may look like the following:

<table>
<thead>
<tr>
<th>Employee</th>
<th>Department</th>
<th>Emergency Contact</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gonzalez, James</td>
<td>Marketing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dawson, John</td>
<td>Finance</td>
<td>Dawson, Jane</td>
<td>555-1212</td>
</tr>
<tr>
<td>Larkins, Abraham</td>
<td>R &amp; D</td>
<td>Taylor, Mary</td>
<td>555-3456</td>
</tr>
<tr>
<td>Walker, George</td>
<td>Finance</td>
<td>Walker, Martha</td>
<td>555-9876</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

NULLS are displayed for employees who do not have emergency contacts.
However, if you model the attributes as described below, you would not get the desired output:

- **Employee**
  - @ID = EMP_ID, @[First Name] = FIRST_NAME, @[Last Name] = LAST_NAME
  - Tables: EMPLOYEE (lookup), EMERGENCY_CONTACT

- **Department**
  - @ID = DEPT_ID
  - Tables: DEPARTMENT (lookup), EMPLOYEE
  - Child: Employee

- **Hire Date**
  - @ID = HIRE_DATE
  - Tables: EMPLOYEE (lookup)
  - Child: Employee

- **Emergency Contact**
  - @ID = CONTACT_PHONE_NUMBER, @[First Name] = CONTACT_FIRST_NAME, @[Last Name] = CONTACT_LAST_NAME
  - Tables: EMERGENCY_CONTACT (lookup)
  - Child: Employee

Using the above model, the SQL generated would join the EMPLOYEE table to the EMERGENCY_CONTACT table, and only those employees who have emergency contacts would appear in the final result. In order to see all employees, you can perform an outer join using a logical view, described as follows.
Using a logical view for outer join

To perform an outer join for the case described above, you can use the following SQL and the list of columns to map to the view:

```sql
select E.EMP_ID,
       E.FIRST_NAME,
       E.LAST_NAME,
       E.HIRE_DATE,
       E.DEPT_ID,
       C.CONTACT_FIRST_NAME,
       C.CONTACT_LAST_NAME,
       C.CONTACT_PHONE_NUMBER
from EMPLOYEE E
left outer join EMERGENCY_CONTACT C
on (E.EMP_ID = C.EMP_ID)
```

Make sure to include all columns from the original child table (for example, EMPLOYEE). The new logical table LVW_EMERGENCY_CONTACT can then be used to define attributes as follows:

- **Employee**
  - @ID = EMP_ID, @[First Name] = FIRST_NAME, @[Last Name] = LAST_NAME
  - Tables: EMPLOYEE (lookup), LVW_EMERGENCY_CONTACT
• **Department**
  - @ID = DEPT_ID
  - Tables: DEPARTMENT (lookup), EMPLOYEE, LVW_EMERGENCY_CONTACT
  - Child: Employee

• **Hire Date**
  - @ID = HIRE_DATE
  - Tables: EMPLOYEE (lookup), LVW_EMERGENCY_CONTACT
  - Child: Employee

• **Emergency Contact**
  - @ID = CONTACT_PHONE_NUMBER, @[First Name] = CONTACT_FIRST_NAME, @[Last Name] = CONTACT_LAST_NAME
  - Tables: EMERGENCY_CONTACT (lookup), LVW_EMERGENCY_CONTACT
  - Child: Employee

The Employee attribute is not represented in the original EMERGENCY_CONTACT table and all attributes represented in the EMPLOYEE table are also represented in the LVW_EMERGENCY_CONTACT table.

Now if we run a report with Employee and Emergency Contact attributes, the EMPLOYEE table will be outer joined to the EMERGENCY_CONTACT table, and NULLs will be returned for any employees who do not have emergency contacts. Also note that if we run a report that includes only the Employee attribute, it will be executed against the EMPLOYEE table; the EMERGENCY_CONTACT table will be joined only when necessary.

This technique is applicable any time an attribute relationship is 0..1, meaning that the lookup tables should always be outer joined. The technique does not work when the lookup tables should sometimes be outer joined and sometimes be inner joined.
Introduction

Data marting is the functional name for a set of capabilities offered by MicroStrategy to store report results in a physical table in a supported relational database.

This chapter describes the concepts you need to know to set up a data marting environment.

Associated terminology

The following are terms associated with data marting:

- **data mart**—a database location, also known as a database instance, dedicated to the storing of report results in the form of relational tables

- **data mart report**—a special kind of report that saves its report data in a database rather than returning those results to the user
• *data mart table*—a table created by a data mart report

Sample business scenarios

Possible data marting applications include

• Application subsetting, which uses MicroStrategy scheduling capabilities to create and periodically refresh lookup and fact tables as subsets of a central data warehouse. For this type of data mart application, projects are built using MicroStrategy Architect.

• Warehouse building, through which you can generate warehouse tables and modify existing schemas to enhance query performance. With this type of application you can, for example,

1. build a report template layout to create a report that yields an aggregated result
2. create a data mart report using the template you have created
3. add the report to a schedule to generate a data mart table
4. add the generated table to the metadata
5. use the schedule created above to refresh the table at either time- or event-driven intervals

• Third party tool integration, which uses ROLAP and MicroStrategy Desktop functionality to extract data from a massive warehouse and build a result table suitable for loading onto third party tools. This capability allows you to create a single table to feed a mass-mailing software tool or a similar closed-loop application, for example.

The primary purpose of data marting is the creation of relational tables that can be used and updated in the same manner as those in a project schema. Data mart tables are created from the information in the columns and rows of reports that are selected or created for this purpose.
To create a relational table for data marting

1. Either create a new report or select an existing one to use for table creation.

2. Using the Report Editor, designate the report as a data mart report.

3. Set the relevant properties, such as table creation location, table name, VLDB properties, governing parameters, and so on, for the report.

4. Execute the report.

5. Resolve any prompts included in the report.

MicroStrategy then creates the data mart table in the database you have selected. When table creation is complete, the system returns a message that includes the table name and notification that table creation was successful.

For table and metric alias naming, ensure that the table and metric alias names follow the naming convention rules for your particular database platform. You will receive an error if you do not use a valid table or metric alias name.

The output of data mart reports: relational tables

When the contents of a report have been used as input for data marting purposes, the result is a table located in a relational database. Data mart tables possess all the characteristics of other data warehouse tables, and have, therefore, the same requirements for creation, including

- A name—This can be any name you want to assign to the table. When the system notifies you that table creation was successful, it includes the table name in the return message. You can select whether the data mart table uses a placeholder as part of the table name.
Placeholders allow you to modify table names dynamically, according to need or convenience. Placeholders available for naming data mart tables are as shown as below.

<table>
<thead>
<tr>
<th>Placeholder</th>
<th>Replacement Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>!U</td>
<td>User login name</td>
</tr>
<tr>
<td>!D</td>
<td>Date on which the table was created</td>
</tr>
<tr>
<td>!O</td>
<td>Report name</td>
</tr>
<tr>
<td>!!</td>
<td>Exclamation (!). &quot;!!=&quot; inserts a &quot;not equal to&quot; sign, (!=) in the SQL statement.</td>
</tr>
</tbody>
</table>

Any character other than D, U, or O, invalidates a placeholder and causes it to be deleted from the data mart table. Placeholder characters are case-sensitive; they should be entered in all-uppercase.

- A location—A data mart table can be located in any relational database. You specify the table location by selecting the database instance in which you want the table to be created. It can be created in the same database as the warehouse or in a different platform altogether. These conditions apply to your selection of data mart table location:
  - If the data mart database is in the same physical location as that of the warehouse, the results do not need to be brought to the server for placement.
  - If the data mart database is in a different physical location from that of the warehouse, the results set needs to be brought to the server for insertion into the new platform.

- VLDB properties—These govern creation parameters for the `CREATE TABLE` statement. The syntax of these statements follows. The pre-SQL statement set is shown with the VLDB properties in italics.

```sql
CREATE <<Table Qualifier>> TABLE <<Table Descriptor>><<Table Prefix>>[TABLE NAME] <<Table Option>> ([COLUMN DEFN]) <<Table Space>> {PRIMARY INDEX/PARTITION KEY/PRIMARY KEY} <<Create Table Post String>>
```
• Column name aliases—Attribute columns can have user-defined names in a data mart table. Attribute column aliases are specified through the Attribute Editor.

• Governing parameters—These specify maximum values for execution time, and for the number of rows to be sent to the Analytical Engine. The maximum row setting applies only to a data mart report that calls the Analytical Engine, such as using the function runningMax in a metric definition.

  The settings at the data mart report level always override the settings at project level. By default, the **Maximum execution time** is set to ‘0’, which means that the project level setting will be used. The project level setting is available in the Governing category of the Project Configuration Editor. For example, if the maximum execution time at data mart level is set to '0', and the setting at the project level is set to 10 seconds, the report will timeout after 10 seconds. But, if you set the maximum execution time as 5 seconds, and the project level setting is specified as 10 seconds, the data mart report level overrides the project level settings, and the report will timeout after 5 seconds.

  Pre- and post-SQL statements allow the application of user-defined SQL before and after the creation of data mart tables. Possible uses include **GRANT** privileges and index creation.

  Pre- and post-table creation settings apply to data mart tables only; they do not affect settings generated to process a report.

---

**Custom groups in data mart tables**

Since custom groups do not exist as columns in the warehouse, when a report includes a custom group the resulting data mart table contains data that does not directly map to corresponding columns in a warehouse table. For this reason, data derived from custom group columns is handled differently.
This figure shows how columns appear in data mart tables created from reports that include custom groups.

<table>
<thead>
<tr>
<th>Elm_ID</th>
<th>Elm_Name</th>
<th>F_ID</th>
<th>F-Name</th>
<th>Band</th>
<th>Metric_Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In a table structured as shown:

- The Element ID column contains the IDs of the custom group elements as generated by the engine.
- The Element Name column contains the descriptions of custom group elements.

The figure that follows shows part of a sample table that includes custom group data.

In this example,

**A** points to the element-ID column as it appears in the data mart table. Element IDs (1, 2, 3,...) are extracted from the corresponding custom group elements in the report.

**B** points to the element names as they appear in the data mart table. These names are extracted from the corresponding custom group element names in the report.
Similarly, when a report includes columns that reflect consolidations, the SQL Engine provides element ID values for the data mart table. In such cases, the table structure looks like the following.

<table>
<thead>
<tr>
<th>Elm_ID</th>
<th>Elm_Name</th>
<th>Metric_Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In a table structured as shown:

- The Element_ID column contains values provided by the engine.
- The Element_Name column contains the names of consolidation elements.

For more information on custom groups, custom group elements, consolidations, and consolidation elements, see Chapter 8, *Custom Groups and Consolidations*. 
Introduction

Transformations are one of many MicroStrategy techniques used to perform time-series analysis, which is a type of analysis relevant to many different industries, including retail, banking, and telco. A typical example of this type of analysis is a TY/LY comparison (This Year versus Last Year). To calculate a variance or a growth percentage, for example, last year’s revenue versus this year’s revenue, it is very convenient to use a transformation even though there are alternatives. Transformations are often the most generic approach and can be reused and applied to other time-series analyses.

This chapter details transformations. It discusses the different types of transformations and how to create and use them.
What is a transformation?

A *transformation* is a schema object that encapsulates a business rule used to compare results of different time periods. Usually defined by a project designer, transformations are used in the definition of a metric to alter the behavior of that metric.

Recall the example used in the *Introduction*, the TY/LY comparison. To calculate this year’s revenue, you can use the Revenue metric in conjunction with a filter for this year. Similarly, to calculate last year's revenue, you can use the Revenue metric in conjunction with a filter for last year. However, a more flexible alternative is to use a previously created Last Year transformation in the definition of a new metric, last year’s revenue. With a single filter, on 2003 for example, the two metrics Revenue and Last Year Revenue will give you results for 2003 and 2002, respectively.

Since a transformation represents a rule, it can describe the effect of that rule for different levels. For instance, the Last Year transformation intuitively describes how a specific year relates to the year before. It can in addition express how each month of a year corresponds to a month of the prior year. In the same way, the transformation can describe how each day of a year maps to a day of the year before. This information defines the transformation and abstracts all cases into a generic concept. That is, you can use a single metric with a last year transformation regardless of the time attribute contained on the report.

The definition of the association between the original value and the transformed one can be represented in an expression that uses columns of the warehouse, constants, arithmetic operators, and mathematical functions. However, it is sometimes desirable to precalculate these values and store them in a table designed for the transformation. This process is sometimes referred to as table-based transformation. The advantage is the possible use of indexing to speed query times, but it requires the management of an additional table in the warehouse. Returning to the TY/LY example, you have the option of using a simple formula such as Year - 1 in the definition of the transformation or precalculating the data and storing it in a column in a table.
Transformation metrics

Transformations are used to compare values at different times. For example, you want to compare sales for this month against sales for the previous month, the same month in the previous year, and so on. Another example is the comparison of year-to-date data against daily sales data. The simple metric tallies daily sales. The transformation metric calculates a rolling total of sales on a daily basis.

While transformations are most often used for discovering and analyzing time-based trends in your data, not all transformations have to be time-based. An example of a non-time-based transformation is This Catalog/Last Catalog, which might use catalog_ID-1 to perform the transformation.

You use transformations to define transformation metrics. A transformation metric is a metric that takes the properties of the transformation applied to it. For example, if you create a metric to calculate total sales and add a Last Year transformation to it, the metric now calculates last year’s total sales.

Any transformation can be included as part of the definition of a metric. Multiple transformations can be applied to the same metric.

Transformations are schema objects and therefore only a project designer with schema object privileges can create them.
Transformation metrics and joint child attributes

In a report, a transformation metric displays the current attribute with transformed data, that is, the values for transformation. For example, a report contains quarter and the transformation metric Last Year’s Revenue. Each quarter is displayed, with the previous year’s revenue, as shown below:

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Metrics</th>
<th>Last Year’s Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 03</td>
<td></td>
<td>1,884,444</td>
</tr>
<tr>
<td>Q2 03</td>
<td></td>
<td>2,585,002</td>
</tr>
<tr>
<td>Q3 03</td>
<td></td>
<td>1,545,399</td>
</tr>
<tr>
<td>Q4 03</td>
<td></td>
<td>2,861,785</td>
</tr>
</tbody>
</table>

When a joint child attribute is added, a conflict arises. The displayed attributes should still be current, with transformed data. However, since the joint child attribute essentially exists in both the time dimension and a non-time dimension, it is not intuitive how the transformation should be performed.

For example, promotion is added to the previous report. The joint child attribute cannot be transformed because not all of its joint children—promotion type and item—are time-related. The report displays the current date, the promotion associated with the current date, and the data from the date-promotion combination, minus one year. A sample report is shown below.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Promotion</th>
<th>Last Year’s Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 2002</td>
<td>Post-Christmas</td>
<td>$1,366</td>
</tr>
<tr>
<td>Q2 2002</td>
<td>Spring Readiness</td>
<td>$483</td>
</tr>
<tr>
<td>Q1 2003</td>
<td>Post-Christmas</td>
<td>$180</td>
</tr>
<tr>
<td>Q1 2003</td>
<td>Valentine’s Day</td>
<td>$99</td>
</tr>
<tr>
<td>Q2 2003</td>
<td>Spring Readiness</td>
<td>$120</td>
</tr>
</tbody>
</table>
Notice that the Valentine’s Day promotion existed in 2003 but not in 2002. While you may want to see it listed for 2002, remember that only the metric values are transformed, not the attributes. That is, since the Valentine’s Day promotion was not run in 2002, the Valentine’s Day-Q1 2002 combination cannot be displayed on the report. No false information is generated.

### Transformation components

All transformations have the following components:

- **Member attributes**—This component contains the attributes to which the transformation applies, that is, the different levels to which the rule applies.

  For example, in the Last Year transformation in the MicroStrategy Tutorial, the member attributes are Year, Quarter, Month, and Day.

- **Member expressions**—Each member attribute has a corresponding expression. In the most generic case, this is an expression that uses constants, arithmetic operators, mathematical functions, and columns from the warehouse, typically the attribute ID column.

  For example, you might create a Last Year transformation using Year_ID-1 as the expression. However, many cases can exist where the data is not conducive to such calculation. If you store Month as 200001 (January 2000), you cannot subtract one and receive December 1999 as the result. A significant advantage to such dynamic calculation is that the database administrator does not have to create and maintain a transformation table. The drawback is that the system must perform the calculation every time.

  There is an important special case where the expression is simply a column from a specific warehouse table specifically populated with data supporting the transformation. The rule is then not encapsulated in an expression but directly in the data of the column. Since the data defines the rule, this approach provides considerable flexibility in the transformation definition. It
is particularly effective when no straight-forward formula can express the rule. In fact, in the case of a one-to-many transformation, a separate table is required. The disadvantage is that the database administrator must create and maintain the additional transformation table in the warehouse. However, once the table is created, it usually significantly decreases the query time.

- **Member tables**—Each member expression is based on a specific table, generally the lookup table corresponding to the attribute being transformed, unless a table was specifically introduced to support this transformation level.

- **Mapping type**—This component determines how the transformation is created based on the nature of the data. The mapping can be one of the following:
  
  - **One-to-one**—A typical one-to-one relationship is “last year to this year.” One day or month this year maps exactly to one day or month from last year.
  
  - **One-to-many**—A typical one-to-many relationship is year-to-date. For one date, there are many other dates included in the year-to-date calculation.

One-to-many transformations can lead to double-counting scenarios. For example, consider YearToDate defined as a one-to-many transformation and Revenue (YTD) as a transformation metric. If this metric is used on a report that does not have Date, which is the member attribute, on the template, and a range of dates is specified in the filter, then the Revenue (YTD) metric will double count.
Example: transformations

Report requirement

You need to analyze your revenue growth quarter to quarter.

How can you accomplish this?

Solution

You can use a transformation metric.

Creating a transformation metric involves two steps. First, the transformation object is defined in the Transformation Editor. Then, the transformation metric needs to be created in the Metric Editor.

The project designer needs to provide his users with a Last Quarter transformation that maps every quarter to the previous one, every month in each quarter to the same month in the previous quarter, and every day of a quarter to the same day in the previous quarter. Refer to the MicroStrategy Tutorial’s example of such a transformation. It is called Last quarter and located in the Transformations subfolder of the Schema Objects folder.

Once the transformation is created, you can use it in a metric to yield last quarter’s revenue. In the Metric Editor, specify the formula for the metric and drag the transformation that you just created into the transformation pane. To calculate the growth quarter to quarter, use the original revenue metric and the new last quarter revenue metric.
The final report contains the quarter attribute and the metrics described above.

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Metrics</th>
<th>Quarterly Revenue</th>
<th>Last Quarter's (Quarterly Revenue)</th>
<th>Quarterly Revenue Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 03</td>
<td></td>
<td>$1,222,374</td>
<td>$2,061,785</td>
<td>($1,639,411)</td>
</tr>
<tr>
<td>Q2 03</td>
<td></td>
<td>$2,382,652</td>
<td>$1,222,374</td>
<td>$1,160,278</td>
</tr>
<tr>
<td>Q3 03</td>
<td></td>
<td>$1,560,244</td>
<td>$2,382,652</td>
<td>($822,408)</td>
</tr>
<tr>
<td>Q4 03</td>
<td></td>
<td>$2,894,534</td>
<td>$1,560,244</td>
<td>$1,334,290</td>
</tr>
</tbody>
</table>
**Aggregate Tables**

**Introduction**

Aggregate tables are summary tables that store data at higher levels than how the data was initially captured and saved. Aggregate tables provide quicker access to frequently examined information, while retaining the traditional power of ROLAP to directly query the database to answer any question.

This chapter describes how and why aggregate tables are used. It builds on your knowledge of fact tables.
Why should I use aggregate tables?

MicroStrategy uses optimized SQL to query the relational database directly to answer users’ questions. Users can therefore ask any question that is supported by the data in their warehouse and then analyze the results until they find their precise answer. The disadvantage to this relational OLAP (ROLAP) methodology is that accessing huge fact tables can be potentially time-consuming. Multidimensional OLAP (MOLAP) was considered by some to be the answer to this problem. However, it is not scalable for large projects because of the difficulty of maintaining every possible combination of aggregates as the number of attributes and the amount of data increases. MicroStrategy’s solution is the use of aggregate tables to provide quicker access to frequently-asked data while still retaining the power to answer any user query.

Some advantages of aggregate tables are as follows:

- reduce input/output, CPU, RAM, and swapping requirements
- eliminate the need to perform dynamic calculations
- decrease the number of physical disk reads and the number of records that must be read to satisfy a query
- minimize of the amount of data that must be aggregated and sorted at run time
- move time-intensive calculations with complicated logic or significant computations into a batch routine from dynamic SQL executed at report run time

In summary, the MicroStrategy SQL Engine in combination with aggregate tables and caching can produce results at about the same speed as MOLAP. But the ability to answer questions on the fly is still retained, and the solution is scalable for large databases, unlike MOLAP.
**Aggregation terminology**

Now that you know why you should use aggregate tables, the following terms and definitions will familiarize you with basic terminology and therefore the basic concepts.

MicroStrategy creates aggregates only on fact tables, since lookup tables and relationship tables are usually significantly smaller. Therefore, to understand aggregate tables, you should be familiar with fact tables in the context of data modeling and data warehousing. For more information on these topics, see Chapter 10, *Facts*, and Appendix D, *Advanced Data Modeling*, as well as the *Data Modeling* appendix in the *MicroStrategy Basic Reporting Guide*.

**Aggregation versus pre-aggregation**

Whenever the display level of data on a report must differ from the level at which it is initially captured, *aggregation*, that is, the rolling up of data, must occur. By default, aggregation occurs dynamically with a SQL statement at report run-time.
For example, sales data is stored by day in a fact table. A report requesting month-level data is executed. The daily values from the fact table are selected, sorted, and added to produce the monthly totals, as shown below.

```
Select Attributes, SUM(Facts) from LU_Table, Fact_Table
Where Join(FactLU)
  Qualifications (Month=March, Store=1, Item=10)
Group by Attributes

<table>
<thead>
<tr>
<th>Revenues</th>
<th>March</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1,305,584</td>
<td></td>
</tr>
</tbody>
</table>
```

Aggregation can also be completed before reports are run, with the results stored in an aggregate table. This process is called pre-aggregation. You can build these pre-aggregated—or aggregate—tables as part of the ETL process. If sales data is frequently requested at the month level, as in the previous example, an aggregate table with the sales data rolled up to the month level is useful. This
eliminates the reading, sorting, and calculation of data from many rows in a large, lower-level fact table at run time. The following diagram illustrates this.

If the daily sales fact table is the lowest-level fact table and contains atomic-level data, it is referred to as a base table. In these terms, an aggregate table is any fact table whose data is derived by aggregating data from an existing base table. It can also be defined as a summary table, storing data at a higher level than how the data was initially captured and saved.
Degree of aggregation

While MOLAP can provide fast performance when it can answer a question, it requires a completely aggregated schema to answer the most questions. That is, every possible combination of aggregate associations must be generated when the multidimensional cube is built. This ensures that all possible questions can be answered. This scenario becomes very difficult to maintain as the number of attributes and the amount of data increase, and therefore is not very scalable.

In a ROLAP environment, the degree of aggregation can be as dense or as sparse as is appropriate for the users. A densely aggregated warehouse has a large number of aggregate tables while a sparsely aggregated warehouse has fewer. Sparse aggregation refers to the fact that a given project only requires as many aggregate fact tables as is useful to its users.

ROLAP, therefore, provides much greater flexibility. Only the aggregate combinations that the project administrator determines are beneficial must be created. That is, if the aggregate table is useful in answering frequently-asked queries, its presence provides a response as fast as a MOLAP system can provide. However, if a certain aggregate combination is rarely or never used, the space in the RDBMS does not need to be consumed and the resources to build that table during the batch process do not need to be used.

When should I use aggregate tables?

Not every attribute level or hierarchy intersection is suitable for pre-aggregation. Build aggregate tables only if they will benefit users, since the creation and maintenance of aggregate tables require additional work by the database administrator. Also, do not waste the database space if the tables will not be used.
Some factors to consider when deciding whether to create aggregate tables:

- the frequency of queries at that level
- the relationship between the parent and child
- the compression ratio

**Frequency of queries at the level**

Build aggregate tables only if they will be useful to your users. If aggregate tables are never accessed, they consume disk space and impose unnecessary burdens on the extraction, transaction, and loading process, as well as the database backup routines. This is in direct contrast to why you create aggregate tables.

However, usefulness is not always easy to quantify. For example, consider the following hierarchy:

```
  Department
     |
     v
  Item
      |
      v
Active Flag
```

The summary of data to the department level seems to be a good candidate for an aggregate table. However, if users frequently want to exclude inactive items, the query must use item-level data and summarize the department data dynamically. Therefore, the department aggregate tables would not be used in this situation.

Once your warehouse is in production, trace the usage of the aggregate tables to determine how frequently they are used in real life. If any table is not used, eliminate it from the warehouse. MicroStrategy Enterprise Manager allows you to easily track table usage. For more information on Enterprise Manager, see the *MicroStrategy System Administration Guide.*
Relationship between the parent and child

When an aggregate table is created, the child records are usually summarized into the parent record, based on the key combinations in a relationship table. In any hierarchical relationship, when the parent-child relationship is altered, all tables that hold that relationship or data relevant to it must be updated. Whether these relationships are dynamic or static change how they are aggregated into tables.

Dynamic relationships

When the relationship between parent and child elements change, the relationship is called dynamic. These changes often occur because of organizational restructuring; geographical realignment; or the addition, reclassification, or discontinuation of items or services. For example, a store can decide to reclassify the department to which items belong.

Aggregate tables that contain dynamic relationships must be recalculated every time a change is made. If the tables are large, this process can take time, consume resources, and complicate the batch process. Frequent changes can mean aggregate tables are not optimal for this situation. Consider the frequency of the changes, the table size, and the impact on the batch process, and then balance the disadvantages against the advantages of having an aggregate table.

Static relationships

When elements rarely or never change relationships, they are termed static relationships. In these cases, maintenance of aggregate tables is very easy. For example, time hierarchies are seldom dynamic—days do not migrate into different weeks, and fiscal weeks do not move into different months.

Also, rolling up an entire hierarchy can avoid many problems with relationship changes. For example, a table contains one value for the sum of all stores. It is not affected by a reorganization within the geography hierarchy.
Compression ratio

The process of data aggregation applies an aggregate function, such as sum or average, to a set of child records to produce a single parent record. The average number of child records combined to calculate one parent record is called the compression ratio. The effectiveness of an aggregate table can be estimated from this number, since it represents the decrease in records that must be read to respond to a query at that level.

Recall that some of the reasons to build aggregate tables are the reduction of disk I/O and the amount of records that must be dynamically sorted and aggregated. Therefore, pre-aggregating data is effective only if the compression ratio is significant. If the compression ratio is 3:2, the aggregate table requires 2/3 of the base table’s storage space but yields only a 1/3 reduction in the number of records. In contrast, if the compression ratio is 4:1, the aggregate table reduces the number of records by 3/4 and uses only 1/4 of the storage space.

When the number of elements differs significantly between two attributes in the same hierarchy, the compression ratio suggests that an aggregate table can provide more efficient queries. Also, for smaller base tables, the resource demands placed on the database server by dynamic aggregations decrease and therefore so does the effectiveness of pre-aggregation. To determine when pre-aggregation is worthwhile for your system, you must balance the importance of speed and the availability of disk space and resources to maintain the schema.

For more information on ratios, refer to the Data Modeling appendix of the Basic Reporting Guide.

Integrating aggregate tables

To integrate an aggregate table into an existing project

1 Add the table to the project in the Warehouse Catalog.
2 Use the new table in the desired fact expressions and attribute form expressions.

If your aggregate table structure is consistent with your base fact table structure, Architect automatically adds it to the definitions of your existing attributes and facts. In other words, Architect is aggregate-aware. How does Architect know to use the aggregate table rather than the base fact table, when either could provide the answer to a query? The answer is logical table size.

**Logical table size**

Architect assigns a size to every table in the project when you first add them to the project. These size assignments are stored in the metadata and are calculated based on the table columns and their corresponding attributes. Because Architect uses the conceptual or logical attribute definitions when assigning sizes, this measurement is known as the logical *table size*.

When you run a report, the Analytical Engine chooses the smallest of all tables, based on logical table size, that contains enough data to answer the query.

**Changing the logical table size**

Remember that the initial logical table size is based on the number of attribute columns and the various levels at which they exist in their respective hierarchies. Suppose the base fact table contains millions of rows of transaction-level detail. The other tables, however, have only higher-level or summary information. Because the attribute levels are lower in the base fact table, the table as a whole is assigned a higher value for the logical table size than are the summary tables with higher-level attributes.

Logically, a table with a higher-level attribute should be smaller in size. Of course, this is not always true in a real warehouse. Therefore, the Logical Table Editor allows you to alter the logical table sizes based on their true relative sizes.
Partition Mapping

Introduction

Partition mapping divides large logical tables into smaller physical tables based on a definable data level, such as month or department. Partitions improve query performance by minimizing the number of tables and records within a table that must be read to satisfy queries issued against the warehouse. By distributing usage across multiple tables, partitions improve the speed and efficiency of database queries.

Time is the most common category for partitioning databases. Partitioning by time limits growth of the database tables and increases stability.

This chapter describes the concepts you need to know before mapping partitions.
Server versus application partitioning

Partitioning can be managed by either the database server or the MicroStrategy application. Either way, the tables are partitioned at the database level—the terms application and server refer to what manages the partitioned tables, not where the tables are split.

Server-level partitioning

In RDBMS server-level partitioning, the database server rather than MicroStrategy manages the partitioned tables. The original fact table is not physically broken into smaller tables. Instead, the database server logically partitions the table according to parameters specified by the database administrator. You do not need to take any action in MicroStrategy to support the partitioning.

Since only the logical table is displayed to the end user, the partitioning is transparent to MicroStrategy. In contrast, in application-level partitioning the relational database is unaware of the partitioned tables.

Refer to your database documentation for details on server partitioning for your particular platform.

Application-level partitioning

In application-level partitioning the application, rather than the RDBMS server, manages the partition tables. A partition base table (PBT) is a warehouse table that contains one part of a larger set of data. Partition tables are usually divided along logical lines, such as time or geography. MicroStrategy supports two types of partitioning:

- Metadata partition mapping stores the mapping information in the project metadata.
- Warehouse partition mapping uses a specialized warehouse table to determine which table to access.
Metadata partition mapping

*Metadata partition mapping* is the mapping of partitions carried out and maintained in the project metadata at the application level. MicroStrategy manages the mapping between the logical table and the physical tables. This design makes it easier for you to specify a flexible partitioning schema.

In a metadata partition mapping, you specify one or more partitioning attributes in the Metadata Partition Mapping Editor. Next you define what attribute elements within those attributes should point to which PBT. You create all of the rules for choosing the appropriate PBT here and the rules are stored in the MicroStrategy metadata.

Homogenous and heterogeneous partitions

Metadata partitions can be homogenous or heterogeneous. With heterogeneous partitioning, the PBTs can have different amounts of data stored in them at different levels. For example, one table can contain six months of sales data, while another stores an entire year. The PBT level, or key, refers to how the data is stored. For example, sales data for the current year can be stored at the daily level, while historical data is saved by month only.

MicroStrategy stores one PBT level for each partition. If all the PBTs within a partition are not stored at the same level, the highest PBT level is used as the PBT level of the partition. For instance, if all the sales data in the previous example is stored in one partition, you will not be able to access current sales at the day level. This is because the PBT level for the partition is month, which is higher than day. If you save current data in a partition at the daily level and the historical data in another, at the month level, you will be able to fully access the data.
In contrast, homogenous partitions must have the same amount of data stored at the same PBT level. The logical structure of the PBTs must be the same, that is, they must have the same facts and attributes defined. To continue with the previous examples, each table must store one year of data at the month level.

Data slices

When you set up metadata partitions, you create the PBTs before defining a data slice. The data slice acts as a filter that describes what portions of data are placed in the partition table. Based on this data slice, the engine knows which table to pick when generating the SQL.

A data slice holds the parameters that a partition is based upon, for example, Month=January. Instead of retrieving data for all months, the server knows to access a particular table that contains the data for January only. By creating a data slice with the partition, you can retrieve specific data without time-consuming joins and searches.

It is very important to create a reasonable and valid data slice because MicroStrategy cannot verify its accuracy or relevance. Thus, the information is known only by you, the project designer. Basically, the data slice must make sense for the data. A poorly crafted data slice can lead to errors from generating incorrect SQL and retrieving the wrong data.

Data slicing displays and can be modified only for the metadata partitioning. Each partition mapping table must include at least one data slice. In a heterogeneous mapping, data slices can exist at different levels and can be composed of different keys.
Attribute qualifications

To create data slices, you use attribute qualifications. Attribute qualifications are types of filters that are applied to attribute forms. These qualifications allow you to limit the type and amount of data that is returned for a report. For example, if you create a report that contains the attribute Country but you want to return only the results for France, you can create a qualification on the attribute Country and use France as the element that appears on the report.

Warehouse partition mapping

Warehouse partition mapping is the mapping of partitions carried out and maintained in the warehouse. You can define a warehouse partition by adding a table with a special structure through the warehouse catalog. This table contains the map for the partition, which is stored in the warehouse. Warehouse partitions divide tables physically along any number of attributes, though this is not visible to the user.

Warehouse partitions must be homogenous, unlike metadata partitions, so that the same amount of data is stored at the same PBT level and the same facts and attributes are defined. Homogenous partitioning divides data of equal levels, like January and February.

The original fact table, which contains all of the data, is not brought into the project. Rather, the database administrator creates multiple smaller physical tables in the data warehouse. Each table contains a subset of the data in the original fact table. The database administrator is responsible for keeping the partitions consistent and up-to-date. He must also create and maintain a partition mapping table (PMT), which is used to identify the partitioned base tables as part of a logical whole.
After the PMT is created, when you run a report in Desktop or Web that requires information from one of the PBTs, the Query Engine first runs a pre-query to the PMT to determine which PBT to access to bring the data back for the report. The pre-query requests the PBT names associated with the attribute IDs from the filtering criteria. When it finds the name of the PBT, it calls the SQL Engine to write the appropriate SQL for the warehouse.

There are no data slices in the warehouse partition.

MicroStrategy supports warehouse partitions on both upgraded and newly created projects. These are added using the Warehouse Catalog Browser.

## Metadata versus warehouse partition mapping

Before MicroStrategy 7, warehouse partition mapping was the only type of application-level partition mapping available. Now you have a second option, metadata partition mapping, which does not require any additional tables in the warehouse. Metadata partition mapping is generally recommended over warehouse partition mapping in MicroStrategy. However, if you already have partition mapping tables that you set up in MicroStrategy 6.x, you can continue to use warehouse partition mapping in MicroStrategy. The basic concepts are similar between the two strategies.

Metadata partition mapping is recommended because you do not need to create and maintain partition mapping tables in the warehouse. You create the rules in MicroStrategy that the Query Engine uses to generate the SQL to run reports. Because you create the partitions directly in the metadata, it is easier to maintain.

Metadata partition mapping also allows both heterogeneous and homogenous partitions, unlike warehouse partition mapping. Only homogenous partitions can be used in warehouse partition mapping.
Introduction

This appendix provides information on the MicroStrategy Tutorial, including the data model and physical warehouse schema.

What is the MicroStrategy Tutorial?

The MicroStrategy Tutorial is a MicroStrategy project (metadata and warehouse are included) and a set of demonstration applications designed to illustrate the rich functionality of the MicroStrategy platform.

A project is the highest-level intersection of a data warehouse, metadata repository, and user community. Conceptually, the project is simply the environment in which all related reporting is done. You create the projects that users access to run reports.
The theme of the MicroStrategy Tutorial project is a retail store for the 2002 - 2003 time period that sells electronics, books, movies and music. The key features include

- Five hierarchies: Customer, Geography, Products, Promotions, and Time. Each hierarchy can be viewed graphically through MicroStrategy Desktop and Web (through documents)

- 10,000 customers and 400,000 items purchased

- Five reporting areas: Human Resources, Inventory, Financial, Product Sales, Supplier

- Options to create reports from MicroStrategy Web or Desktop focusing on a particular analysis area, such as Customer, Inventory, Time, Products, Category, Employee, or Call Center

**MicroStrategy Tutorial reporting areas**

As noted above, the analysis areas are grouped into five report categories that illustrate the various types of business analysis possible with MicroStrategy:

- Financial—Reports containing information based on time, geography, and products, such as Regional and Quarterly Profit Margins.

  The Financial Reports represent the types of financial reports used in any business. These reports include profit and loss information, company forecasts, and margin reports. These reports give executives, general managers, and operations managers immediate access to financial data so that they can quickly analyze trends and key performance indicators. They ensure that all decision-makers have access to a single repository of financial information, so executives can be sure that departments are all working from the same set of facts. Decision-makers are able to determine immediately the profitability of categories, departments, districts, and business units. Individual managers are able to determine their own performance against budget plan and standard
business performance metrics. Furthermore, decision-makers can get timely reports on key metrics, uncover opportunities to raise revenue and lower costs, track changes in operational costs, analyze categories and business units, and compare actual performance against budget.

- **Human Resources**—Reports containing information on employees; headcount, birthdays, length of employment, top 5 employees by revenue. These reports are based on employees, time, geography, and sales.

  The Human Resources Reports provide insight into human capital so that managers can boost the efficiency and effectiveness of their employees. Human Resource Representatives can highlight under-performing employees and misallocated headcount. Managers at all levels can focus on the performance of their people, drill down to an individual employee detail level, view trends, and extract intelligence not otherwise evident.

- **Inventory**—Reports containing information based on supplier, product, cost, and profit, such as Inventory and Unit Sales, or Inventory Received from Suppliers by Quarter.

  The Inventory Reports track inventory information within the company and through to suppliers. Essentially these reports show how many units of an item are on hand, how many are expected from a particular supplier, and units sold. Inventory reports are used to ensure that the supply chain is as efficient as possible. Using these reports, employees can analyze trends and details, quickly adjust inventory and distribution, and understand underlying supply chain costs and inefficiencies.

- **Product Sales**—Reports that allow for market basket analysis, such as Sales by Region, Revenue over Time, and Yearly Revenue Growth by Customer Region.

  The Product Sales Reports allow managers and analysts to monitor and analyze sales trends, track corporate revenue goals, compare store-to-store performance, and respond more quickly and accurately to feedback from the marketplace. In turn, executives can analyze sales trends and details, quickly adjust pricing and promotions, identify product affinities, key profit centers, and understand costs and revenue trends.
• Supplier—Reports containing supplier, sales, profit, and revenue information, such as Brand Sales by Supplier, Supplier Sell-Through Percentage, and Units Sold and Profit by Supplier.

The Supplier Reports allow managers and analysts to monitor and analyze vendor performance so that they can quickly identify performance problems. These reports track brands and items sold that came from a particular vendor. They also correlate profit and revenue information with particular suppliers so that relationships with key vendors can be strengthened.

These reports are located in the Reports folder of the MicroStrategy Tutorial project.

Once the areas of analysis are determined, a data model is created.

The MicroStrategy Tutorial data model

A logical data model graphically depicts the flow and structure of data in a business environment. It provides a way of organizing facts so that they can be analyzed from different business perspectives. For example, a simple logical data model for a retail company might organize all necessary facts by store, product, and time—three common business perspectives typically associated with retail business.

For more detailed information about data modeling, refer to the Data modeling appendix in the Basic Reporting or Introduction to MicroStrategy documents.

For the purpose of the MicroStrategy Tutorial, the areas of analysis discussed earlier, Financial, Product Sales, Human Resources, and so on, are organized into the following hierarchical groupings:

• geography
• products
• customers
• time
• promotions

These MicroStrategy Tutorial hierarchies are displayed on the following pages for your reference.

**Data modeling notations**

The following notations are used in the graphical depictions of the following hierarchies:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Indicates</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>entry point</td>
<td>An entry point is a shortcut to an attribute element in the Data Explorer. Creating an entry point grants you faster access to the attribute without having to browse through multiple attributes to reach different levels of the hierarchy.</td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td>A data level defined by the system architect and associated with one or more columns in the data warehouse lookup table. Attributes include data classifications like Region, Order, Customer, Age, Item, City, and Year. They provide a handle for aggregating and filtering at a given level.</td>
</tr>
<tr>
<td></td>
<td>one-to-many</td>
<td>An attribute relationship in which every element of a parent attribute relates to multiple elements of a child attribute, while every element of the child attribute relates to only one element of the parent. The one-to-many attribute relationship is the most common in data models.</td>
</tr>
</tbody>
</table>
Geography hierarchy

The MicroStrategy Tutorial Geography hierarchy contains attributes such as Country and Region, as well as Distribution Center, Call Center, and employee-specific attributes. It might be easy to understand why Country and Region are in the Geography hierarchy, but what about Distribution Center, Call Center, and the employee-related attributes?

The data used in MicroStrategy Tutorial is based upon a fictitious company that sells electronics, movies, music and books. The company does not have physical stores, but instead does its business from catalog and Web sales. Customers review the products in a printed or online catalog and call in their order over the phone. The order is then processed by an employee located at one of the call centers. The order is then fulfilled by a distribution center that holds the correct item and sends it via one of the shippers.

The Geography hierarchy contains the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>Countries where the company does or hopes to do business in the future. Also, Countries where employees work.</td>
<td>USA, Spain, France</td>
</tr>
<tr>
<td>Region</td>
<td>Each country is split into Regions.</td>
<td>Central, Northeast, Southwest</td>
</tr>
<tr>
<td>Call Center</td>
<td>Where product phone-in orders are taken. Each Call Center is located in a different city.</td>
<td>Atlanta, Boston, Charleston</td>
</tr>
<tr>
<td>Distribution Center</td>
<td>The location where product orders are sent out to customers. Currently, each is located in the same city as the Call Center it services.</td>
<td>Miami, New Orleans, Fargo</td>
</tr>
<tr>
<td>Manager</td>
<td>Person responsible for a specific Call Center</td>
<td>Peter Rose, Alice Cooper</td>
</tr>
<tr>
<td>Employee Experience</td>
<td>The number of years an employee has worked for the organization</td>
<td>3, 5, 6</td>
</tr>
<tr>
<td>Hire Date</td>
<td>The date on which a particular employee was hired</td>
<td>2/16/2002, 3/15/2003</td>
</tr>
<tr>
<td>Salary</td>
<td>The amount of money an employee makes per year</td>
<td>24,000, 35,000</td>
</tr>
<tr>
<td>Employee Age</td>
<td>The age of each employee</td>
<td>29, 36, 52</td>
</tr>
</tbody>
</table>
Refer to the following graphic to see how all these attributes are organized into the MicroStrategy Tutorial Geography hierarchy.
Products hierarchy

The products hierarchy contains attributes such as category, brand, catalog, and supplier. It should be noted that the attributes Transaction, Warranty, and Discontinued Code are not part of the main data model—these are extra attributes that were introduced to support the MicroStrategy Narrowcast Server demos.

The Products hierarchy contains the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Products are organized into Categories at the highest level.</td>
<td>Electronics, Music</td>
</tr>
<tr>
<td>Subcategory</td>
<td>Used to further differentiate a subset of Products within a Category</td>
<td>Business, Cameras, Drama</td>
</tr>
<tr>
<td>Warranty</td>
<td>The time period in months during which a manufacturer repairs a broken item (specific to Narrowcast Server)</td>
<td>3, 5</td>
</tr>
<tr>
<td>Brand</td>
<td>The manufacturer or artist for a particular product</td>
<td>Ayn Rand, 3Com, Sony</td>
</tr>
<tr>
<td>Catalog</td>
<td>The medium used to sell products</td>
<td>Spring 2002, Fall 2003</td>
</tr>
<tr>
<td>Supplier</td>
<td>The distributor for a set of Brands</td>
<td>McGraw Hill, Disney Studios</td>
</tr>
<tr>
<td>Discontinued Code</td>
<td>(Currently not implemented in the project.) 0 = discontinued product, 1 = non-discontinued product.</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>The individual Product sold</td>
<td>The Great Gatsby, Sony Discman</td>
</tr>
<tr>
<td>Transaction</td>
<td>Describes a resupply transaction from the fictitious company that the MicroStrategy Tutorial product uses to its suppliers for additional stock</td>
<td></td>
</tr>
</tbody>
</table>

Refer to the following graphic to see how all these attributes are organized into the MicroStrategy Tutorial Products hierarchy.
Customers hierarchy

The Customers hierarchy contains customer demographic and purchase information, such as Customer Age, Income Bracket, Payment Method, and Ship Date.

The Customers hierarchy contains the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Region</td>
<td>The highest level of differentiation for where Customers live</td>
<td>Northeast, South, France</td>
</tr>
<tr>
<td>Customer State</td>
<td>Each Customer Region is divided into multiple States.</td>
<td>Main, North Dakota</td>
</tr>
<tr>
<td>Customer City</td>
<td>Each Customer State is broken down into Cities</td>
<td>Albany, Chicago, Memphis</td>
</tr>
<tr>
<td>Customer Age</td>
<td>The Age of a particular customer at a current point in time</td>
<td>26, 38, 59</td>
</tr>
<tr>
<td>Customer Birth Date</td>
<td>The Date on which the Customer was born</td>
<td>8/4/50, 4/30/72</td>
</tr>
<tr>
<td>Income Bracket</td>
<td>The salary range reported by the Customer</td>
<td>$31,000 - 40,000, $61,000 - 70,000</td>
</tr>
<tr>
<td>Zip Code</td>
<td>The lowest level of differentiation for where Customers live</td>
<td>07026, 36303</td>
</tr>
<tr>
<td>Customer</td>
<td>The name of the individual Customer</td>
<td>Selene Allen, Chad Laurie</td>
</tr>
<tr>
<td>Shipper</td>
<td>The vendor used to send Products to the Customer</td>
<td>Pronto Packages, MailFast</td>
</tr>
<tr>
<td>Rush Order</td>
<td>(Currently not implemented in the project.) Indicates whether a customer chose to expedite delivery of an Order</td>
<td></td>
</tr>
<tr>
<td>Payment Method</td>
<td>The way a Customer pays for an Order</td>
<td>Amex, Check</td>
</tr>
<tr>
<td>Ship Date</td>
<td>The Date on which an Order is shipped from the Distribution Center</td>
<td>9/15/02, 3/26/03</td>
</tr>
<tr>
<td>Order</td>
<td>The tracking number associated with a particular group of Items purchased</td>
<td>167, 2635</td>
</tr>
</tbody>
</table>
Refer to the graphic below to see how all these attributes are organized into the MicroStrategy Tutorial Customers hierarchy.
Time hierarchy

The Time hierarchy contains time-specific attributes—Year, Quarter, Month, and Day.

The Time hierarchy contains the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>Calendar Year of purchase.</td>
<td>2002, 2003</td>
</tr>
<tr>
<td>Quarter</td>
<td>Calendar Quarter of purchase.</td>
<td>Q2 02, Q3 03</td>
</tr>
<tr>
<td>Month of Year</td>
<td>Calendar Month of purchase.</td>
<td>January, November</td>
</tr>
<tr>
<td>Month</td>
<td>Month of purchase.</td>
<td>Jul 02, Aug 03</td>
</tr>
<tr>
<td>Day</td>
<td>Calendar Date of purchase.</td>
<td>5/14/02, 12/26/03</td>
</tr>
</tbody>
</table>

Refer to the graphic below to see how all these attributes are organized into the MicroStrategy Tutorial Time hierarchy.
**Promotions hierarchy**

The Promotions hierarchy contains Promotion and Promotion Type. This hierarchy is useful for recording whether a sale was a promotional purchase.

The Promotions hierarchy contains the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Promotion Type</td>
<td>(Currently not implemented in the project.) Type of discount period offered (Sale type).</td>
<td>Mother’s Day, Labor Day</td>
</tr>
<tr>
<td>Promotion</td>
<td>(Currently not implemented in the project.) Date range for a particular discount period under which an Item is purchased (Sales Date).</td>
<td>9/1/02 - 9/4/02, 2/16/03 - 2/19/03</td>
</tr>
</tbody>
</table>

Refer to the graphic below to see how all these attributes are organized into the MicroStrategy Tutorial Promotions hierarchy.

![Diagram of the Promotions hierarchy]

**Viewing the MicroStrategy Tutorial data model**

Although the MicroStrategy Tutorial data model is displayed in the previous pages, you can also view it directly in the product.
To view the MicroStrategy Tutorial data model

1. If you are not already using the Tutorial, log in to the project source containing the MicroStrategy Tutorial and expand the MicroStrategy Tutorial project. You must log in as an Administrator (user name Administrator, no password) to complete these steps.

2. From the Schema menu, point to **Graphical View**, and then choose **Hierarchies**. Once loaded, the Hierarchies - MicroStrategy Tutorial dialog box opens.

3. To view a different hierarchy, select it from the Hierarchy drop-down menu in the toolbar.

4. To focus on a different entry point, select it from the Entry Point drop-down menu in the toolbar.

5. To view the entire hierarchy in the window, click **Fit in window** from the toolbar.

6. You can rearrange the attributes by dragging and dropping them.

   ![This does not affect the browse order, but allows you to view the hierarchy in a way meaningful to you.]

7. To return to the default view, click **Auto arrange** in the toolbar.

8. To save the layout view of the hierarchy, click **Save** in the toolbar. The next time you open the Hierarchy Viewer, this saved view is displayed.

Once the data model is created, the next step is the schema.
The MicroStrategy Tutorial schema

A schema is a logical and physical definition of warehouse data elements, physical characteristics, and interrelationships.

The logical data model is a picture of all the pieces of information necessary to understand your data and how it relates to your business. It is a graphic-intensive technique that results in a data model representing the definition, characteristics, and relationships of data in a business, technical, or conceptual environment.

The physical warehouse schema is based on the logical data model, such as Day, Item, Store, or Account. Several physical warehouse schemas can be derived from the same logical data model. While the logical data model tells you what facts and attributes to create, the physical warehouse schema tells you where the underlying data for those objects is stored. The physical warehouse schema describes how your data will be stored in the data warehouse.

This appendix shows the physical warehouse schema with datatypes shown.

For more detailed information on the schema, refer to the Data modeling appendix in the Basic Reporting or Introduction to MicroStrategy documents.

The MicroStrategy Tutorial schema is divided into the following parts:

- geography
- products
- customers
- time
- promotions
- fact tables
Schema notations

The following notations are used in the graphical depictions of the following MicroStrategy Tutorial schema.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Indicates</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU_</td>
<td>a lookup table</td>
<td>A database table used to uniquely identify attribute elements. They typically consist of descriptions of dimensions. Lookup tables are usually joined to fact tables in order to group the numeric facts in the fact table by dimensional attributes in the lookup tables.</td>
</tr>
<tr>
<td></td>
<td>a primary key</td>
<td>In a relational database, the set of columns required to uniquely identify a record in a table.</td>
</tr>
<tr>
<td>REL_</td>
<td>a relationship table</td>
<td>While lookup tables store information about one or more attributes, relate tables store information about the relationship between two attributes. Relate tables contain the ID columns of two or more attributes, thus defining associations between them.</td>
</tr>
<tr>
<td>PMT_</td>
<td>a partition mapping table</td>
<td>A warehouse table that contains information used to identify the partitioned base tables as part of a logical whole. Also referred to as a PMT.</td>
</tr>
</tbody>
</table>

The schema also contains fact tables. A fact table is a database table containing numeric data that may be aggregated along one or more dimensions. Fact tables may contain atomic or summarized data. The basic facts from which all metrics in the MicroStrategy Tutorial were created from are listed below:

<table>
<thead>
<tr>
<th>Fact</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>The total amount charged by the supplier to the company</td>
</tr>
<tr>
<td>Discount</td>
<td>A monetary reduction made from a regular price</td>
</tr>
<tr>
<td>End on hand</td>
<td>The number of individual items remaining at the close of each month</td>
</tr>
<tr>
<td>Freight</td>
<td>The compensation paid for the transportation of goods</td>
</tr>
<tr>
<td>Profit</td>
<td>The excess of the selling price of goods over their cost</td>
</tr>
<tr>
<td>Revenue</td>
<td>The total income produced by a given source accounting for all product sales deducting discounts</td>
</tr>
<tr>
<td>Rush Charge</td>
<td>The amount of money charged to expedite delivery service</td>
</tr>
<tr>
<td>Unit Cost</td>
<td>The amount of money charged by the supplier to the company per individual item purchased</td>
</tr>
</tbody>
</table>
### Fact & Description

<table>
<thead>
<tr>
<th>Fact</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Price</td>
<td>The amount of money charged by the company to the customer per individual item sold</td>
</tr>
<tr>
<td>Unit Profit</td>
<td>Unit price - unit cost</td>
</tr>
<tr>
<td>Units Received</td>
<td>The number of individual items acquired from a supplier</td>
</tr>
<tr>
<td>Units Sold</td>
<td>The number of individual items bought by customers</td>
</tr>
</tbody>
</table>

### Geography schema

![Geography schema diagram](image-url)
Products schema
Customers schema
Time schema

Promotions schema
Sales fact tables
Inventory fact tables

- **INVENTORY_ORDERS**
  - MONTH_ID
  - ITEM_ID
  - UNITS_RECEIVED
  - MONTH_DURATION

- **PMT_INVENTORY**
  - QUARTER_ID
  - PBRTNAME

- **INVENTORY_Q1_2000**
  - MONTH_ID
  - ITEM_ID
  - BOH_QTY
  - EOH_QTY

- **INVENTORY_Q2_2000**
  - MONTH_ID
  - ITEM_ID
  - BOH_QTY
  - EOH_QTY

- **INVENTORY_Q3_2000**
  - MONTH_ID
  - ITEM_ID
  - BOH_QTY
  - EOH_QTY

- **INVENTORY_Q4_2000**
  - MONTH_ID
  - ITEM_ID
  - BOH_QTY
  - EOH_QTY

- **INVENTORY_Q1_2001**
  - MONTH_ID
  - ITEM_ID
  - BOH_QTY
  - EOH_QTY

- **INVENTORY_Q2_2001**
  - MONTH_ID
  - ITEM_ID
  - BOH_QTY
  - EOH_QTY

- **INVENTORY_Q3_2001**
  - MONTH_ID
  - ITEM_ID
  - BOH_QTY
  - EOH_QTY

- **INVENTORY_Q4_2001**
  - MONTH_ID
  - ITEM_ID
  - BOH_QTY
  - EOH_QTY

Miscellaneous fact tables

- **RUSH_ORDER**
  - ORDER_ID
  - RUSH_CHARGE

- **PROMOTIONS**
  - ITEM_ID
  - DAY_DATE
  - PROMO_SALE_ID
  - DISCOUNT

- **INVENTORY_CURR**
  - ITEM_ID
  - TARGET_QTY
  - EOH_QTY
  - ON_ORDER_QTY
  - UNIT_COST
  - REORDER_QTY
  - TOTAL_AMT
  - LAST_TRANS_ID
Viewing the MicroStrategy Tutorial schema

Although the MicroStrategy Tutorial physical schema is displayed in the previous pages, you can also view it or the logical schema directly in the product.

To view the MicroStrategy Tutorial schema

1. If you are not already using the Tutorial, log in to the project source containing the MicroStrategy Tutorial and expand the MicroStrategy Tutorial project. You must login as an Administrator (user name Administrator, no password) to complete these steps.

2. From the Schema menu, point to Graphical View, and then choose Tables. Once loaded, the Tables - MicroStrategy Tutorial dialog box opens with the physical view displayed.

3. To switch to the logical view, select View, then Logical View.

4. To change display preferences for the physical view, use the following from the Options menu:
   - Show joins: Select whether to connect the tables to represent the joins between the warehouse tables.
   - Use circular joins: Select whether to use circular joins.
   - Show column data types: Select whether to show the data type and size for each column.
   - Show table prefixes: Select whether to display the table prefix as part of the table name.
5 To change display preferences for the logical view, use the following from the Options menu:
   – Show joins: Select whether to connect the tables to represent the joins between the table columns.
   – Use circular joins: Select whether to use circular joins.
   – Show relationships: Choose whether to map the relationships between the tables.
   – Show relationship types: Choose whether to differentiate between one-to-one, one-to-many, many-to-one, and many-to-many relationships.
   – Show columns: Select whether to display the warehouse columns that define each attribute, as a link between the logical and physical views.

6 To switch back to the physical view, select View, then Physical View.

7 To view the entire schema in the window, click Fit in window from the toolbar.

8 You can rearrange the tables by dragging and dropping them.

   This does not affect the relationships or joins, but allows you to view the tables in a way meaningful to you.

9 To return to the default view, click Auto arrange in the toolbar.

10 To save the layout view of the tables, click Save in the toolbar. The next time you open the Table Viewer, this saved view is displayed.

11 To copy the layout view, select Copy as Metafile from the File menu.
Data Types

Description

Every column used in MicroStrategy is associated with a MicroStrategy data type. The MicroStrategy data type is used during SQL generation when defining intermediate tables, datamart tables, and generating correct syntax for literals. The data type is also used to store data values internally and in the metadata repository.

This appendix describes the data types that MicroStrategy supports, including what they are and how to use them.
Mapping data sources to MicroStrategy data types

To generate SQL or retrieve data from data sources, the MicroStrategy program must be aware of the data types that exist in the database. As each RDBMS supports a different set of data types, the MicroStrategy program generalizes them into a set of MicroStrategy data types. When the database catalog is read from the Warehouse Catalog, all columns in the database are automatically mapped to a MicroStrategy data type.

If Warehouse Catalog Editor displays a column with data type as Unknown, it implies that the data type in the database has not mapped to one of the MicroStrategy data types.

The MicroStrategy data types are listed in the following table:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big Decimal</td>
<td>Represents high-precision fixed point numbers.</td>
</tr>
<tr>
<td>Binary</td>
<td>Represents fixed-length bit strings.</td>
</tr>
<tr>
<td>Char</td>
<td>Represents fixed-length character strings.</td>
</tr>
<tr>
<td>Date</td>
<td>Represents calendar dates.</td>
</tr>
<tr>
<td>Decimal</td>
<td>Represents fixed point numbers up to 15 digits of precision.</td>
</tr>
<tr>
<td>Double</td>
<td>Represents 8-byte floating point numbers.</td>
</tr>
<tr>
<td>Float</td>
<td>Represents 4-byte floating point numbers.</td>
</tr>
<tr>
<td>Integer</td>
<td>Represents signed integer values.</td>
</tr>
<tr>
<td>LongVarBin</td>
<td>Represents large strings of bits.</td>
</tr>
</tbody>
</table>
### Format types

Attribute forms are also associated with a format type, which specifies how attribute form values should be displayed on the MicroStrategy interface. The format types are:

- **Date** - Stores dates in a sequential form to perform calculations on the dates. It represents dates in the MM/DD/YYYY format.

- **Datetime** - Information is displayed both as date and time in the format specific to the data. The date follows the MM/DD/YYYY format and time follows the HH:MM:SS format.

- **Email** - Information is stored in the form of an e-mail address.

- **HTML Tag** - Information is stored as an HTML tag.

- **Number** - Information is displayed in a number format.

### Data Type Table

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LongVarChar</td>
<td>Represents large strings of characters. Similar to ANSI CLOB.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Represents fixed point numbers up to 15 digits of precision. Similar to ANSI NUMERIC.</td>
</tr>
<tr>
<td>Real</td>
<td>Represents 4-byte floating point numbers. Similar to ANSI REAL.</td>
</tr>
<tr>
<td>Time</td>
<td>Represents time of day. Similar to ANSI TIME.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Represents combinations of calendar date and time of day. Similar to ANSI TIMESTAMP.</td>
</tr>
<tr>
<td>Unsigned</td>
<td>Represents unsigned integer values.</td>
</tr>
<tr>
<td>Varbin</td>
<td>Represents variable-length bit strings. Similar to ANSI BIT VARYING.</td>
</tr>
<tr>
<td>Varchar</td>
<td>Represents variable-length character strings. Similar to ANSI VARCHAR.</td>
</tr>
</tbody>
</table>
- **Picture** - Information is stored in the form of an image file, such as bitmap, JPG, or gif.

- **Text** - Information is stored and displayed in the text format.

- **Time** - Represents time in the HH:MM:SS format. This displays only the time and not the date.

- **URL** - Information is stored as either an absolute or a relative Universal Resource Locator.

### Big Decimal

Big Decimal is a MicroStrategy-specific data type that allows users to support high-precision attribute ID values that have more than 15 digits of precision, such as BIGINT and DECIMAL (precision, scale) data types. Examples of such attribute ID values are account numbers, credit card numbers, and long integers.

### Using the Big Decimal data type

With the Big Decimal data type, MicroStrategy preserves the precision of attribute ID values and attribute ID forms when displaying IDs and performing operations such as filtering, drilling, and page by. You can define attributes that are identified by numeric columns in the database. These numeric columns can have more than 15 digits of precision, such as account numbers, and other long integers. You must use the Big Decimal data type to handle these values, because these data values have higher precision and cannot be stored in normal numeric data types.

If you do not associate high-precision database fields with the Big Decimal data type, you may see numbers truncated starting from the 16th digit. The WHERE clause in the report SQL statement in drill reports may truncate numbers starting from the 16th digit, and page by may not return results.
When using the Big Decimal data type, make sure to follow the rules listed below:

- **Constant**: You can force a constant to be stored as a Big Decimal value by enclosing it in hash marks. For example, you can define a filter as "Customer@ID exactly #12345678#", even though 12345678 do not necessarily require the Big Decimal data type.

- **Attribute form**: On the column alias tab in the Attribute Form dialog box, you must change both the form format and the column data type to Big Decimal.

- **Attribute ID**: Follow the steps in the topic *Defining attributes with high-precision ID forms* in the MicroStrategy Desktop online help.

- **Metric**: Though it is possible to define Big Decimal as the data type for metric values, you need to heed the following drawbacks:
  - Precision is lost when any Analytical Engine calculation is performed.
  - Precision is lost when the metric is used in the calculated field in a document.
  - Precision is lost when subtotals on the metric are performed.
  - Precision is lost when metric values are displayed in Graph view.
  - Number formatting strings are not supported on the Web.
  - Some number formatting strings are not supported on MicroStrategy Desktop.
  - When qualifying on a Big Decimal metric, you must explicitly identify high-precision constants by enclosing the value within hash (#) symbols. For example, #1234567890123456#.
Note that the Warehouse Catalog Editor does not automatically map DECIMAL(p, s) or NUMERIC(p, s) columns to the Big Decimal MicroStrategy data type even when the precision is greater than 15. This is because Big Decimal should only be used when the column is used as an attribute ID form.
PASS-THROUGH EXPRESSIONS

Description

Pass-through expressions, also called Apply functions, provide access to functionality that is not standard in MicroStrategy products but can be obtained through the relational database. Pass-through expressions act as containers for non-standard SQL expressions that MicroStrategy does not support. The MicroStrategy Engine recognizes these containers as holding information. When you use an apply function to define an expression in an attribute form, fact, filter, metric, transformation and so on, the SQL Engine essentially ignores the SQL within the apply function. It passes it to the relational database as written.

It is important to understand that while an Apply function can be used wherever the function type it belongs to is applied, you should NOT use any Apply functions when standard MicroStrategy functions can be used to achieve the goal. The Apply functions were not created to take the place of the standard MicroStrategy functions. Instead, they were created to enhance the MicroStrategy product by taking advantage of what the RDBMS platforms can offer.
Use an Apply function ONLY when support does not exist in the MicroStrategy product, because using Apply functions effectively bypasses the validations and other benefits of the product. In the meantime, kindly submit an enhancement request so that MicroStrategy can evaluate your needs for inclusion in a future product release.

This appendix describes the Apply functions, including what they are and how to use them.

The Apply functions

The Apply functions are intended to provide access to the special functions or syntactic constructs that are not standard in MicroStrategy, but are offered by various relational database management system (RDBMS) platforms.

There are five predefined Apply functions, each belonging to a different function type:

- **ApplySimple**: belongs to the single-value function type that includes arithmetic operators, Abs, Round, and so on
- **ApplyAgg**: belongs to the group-value function type that includes Avg, Sum, Min, Max, and so on
- **ApplyOLAP**: belongs to the OLAP function type that includes RunningSum, Ntile, Rank and so on
- **ApplyComparison**: belongs to the comparison function type that includes greater than (>), between, equal (=) and so on
- **ApplyLogical**: belongs to the logical function type that includes Or, And, and Not

For more information on the five types of functions, refer to the *MicroStrategy Analytical Functions Reference*. 
Function syntax

The syntax for these functions contains argument markers flagged by # characters. The argument markers contained within the # flags are replaced in the engine by the actual expressions.

The syntax for the Apply functions is as follows:

```
ApplyFUNNAME("expression_with_placeholders", argument_1, ..., argument_n)
```

The placeholders are represented by #0, #1, and so on. “#” is a reserved character for MicroStrategy, and “n” is the number of the arguments outside the quotes, starting with “0” and increasing in increments of “1.” For example:

```
ApplyComparison("ComparisonFunction(#0,#1)", attribute1@ID, attribute2@ID)
```

Constants are inserted as they appear, and object names are handled depending upon their type. The functions represent custom database-specific functions or other custom SQL. Since the functions are not analyzed or validated by the engine, there are no criteria on what they could possibly contain, as long as the result returned is compatible with what the Analytical Engine expects.

To use # as a character rather than a placeholder, use four # characters in a row. For example, the formula for an Apply function is written as the following:

```
ApplyComparison(UPPER(#0) like 'Z####%', Country@DESC)
```

The SQL for the function is:

```
Select a.11[COUNTRY_ID] AS COUNTRY_ID from [LU_COUNTRY] all
where upper(a11.[COUNTRY_NAME])
like 'Z#'
```
Argument types

The number of allowable arguments is variable. The engine does not verify arguments until the parameter markers are replaced at parsing. Parsing occurs when either OK or Validate is clicked in an expression editor. At parsing time, the engine searches for acceptable argument types. Acceptable argument types are names of MicroStrategy object types or an argument that contains a name of MicroStrategy object types.

Upgrading database types

When upgrading the database type of a project, the custom SQL expression may need to be converted. If the upgrade is a simply upgrade from one version of a database platform to a new version, the Apply function most likely will not need to be changed. However, you should check the documentation of the new version to insure that any SQL syntax changes in the new database version do not affect your Apply functions.

Changing database types

If you are changing database types for your project, you must change all Apply functions to use the correct syntax for the new database platform. For example, if you change your project from an Oracle database type to a DB2 database type, then your Apply functions mostly likely are using Oracle specific SQL syntax and now need to use DB2 specific syntax. The following table shows how the syntax can change across different database platforms.
Refer to your specific database syntax when preparing pass-through expressions. This MicroStrategy Tutorial example shows different expressions for three types of relational databases to determine customer age.

<table>
<thead>
<tr>
<th>Warehouse Type</th>
<th>Expression</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Server</td>
<td>ApplySimple(&quot;datediff(YY,#0, getdate())&quot;, Cust_Birthdate)</td>
<td>…Where Datediff(YY,'06/21/74',getdate())…</td>
</tr>
<tr>
<td>Oracle</td>
<td>ApplySimple(&quot;ROUND(MONTHS_BETWEEN(SYSDATE,#0)/12,0)&quot;, Cust_Birthdate)</td>
<td>…Where ROUND(MONTHS_BETWEEN(SYSDATE, '06/21/74')/12,0)…</td>
</tr>
<tr>
<td>DB2</td>
<td>ApplySimple(&quot;ROUND((days(current date)-days(#0))/365,0)&quot;, Cust_Birthdate)</td>
<td>…Where ROUND((days(current date)-days('06/21/74')/365,0)…</td>
</tr>
</tbody>
</table>

Syntax examples

The examples in the following sections are specific for different databases. Please check with your database documentation to see what SQL syntax is correct for you.

ApplySimple

You can use any database-specific functions or simple operator in ApplySimple functions to apply them directly in the SQL.

In general, ApplySimple can be used to create the following objects:

- attribute form

For any Apply function, the attribute form in the arguments should be a single form, not a form group. The engine ignores any definitions based on attribute forms.
• consolidation
• custom group
• fact
• metric
• subtotal
• transformation

Examples: Good

• `ApplySimple("Datediff(YY,#0,getdate())", [BIRTH_DATE])`
• `ApplySimple("Months_between(sysdate,#0)", [CURRENT_DT])`

Examples: Bad

• `ApplySimple("Sum(#0)", [Column 1])`
• `ApplySimple("Count(#0)", [Column 2])`

The above two examples are bad and should be not used in your application because of the following two reasons:

• `ApplySimple` is a single-value functions and therefore can ONLY be used with single-value functions. Sum and Count are both group-value functions and therefore should not be used with `ApplySimple`.

• Sum and Count are both supported by MicroStrategy and not database-specific; therefore, they should not be used with `ApplySimple` or any other `Apply` functions.

**ApplyAgg**

ApplyAgg is used to define metrics or subtotals by using any database-specific group-value functions (usually the ones that are not recognized as a MicroStrategy object). It accepts facts, attributes, and metrics as input. Although this function allows you to send commands straight to your database, it is important to be aware that the function itself requires proper syntax in order to be a valid expression.
Example:

```sql
ApplyAgg("Regrsxx(#0, #1)", [Fact 1], [Fact 2])
```

### ApplyOLAP

ApplyOLAP was previously called ApplyRelative. If you are working with a project that has not been upgraded to 7i you may still see the ApplyRelative name. Like ApplySimple, it is used to define metrics. However, the main difference between ApplySimple/ApplyAgg and ApplyOLAP is that ApplyOLAP only accepts metrics as input since it is used with OLAP functions such as Rank. It is usually used with any database-specific OLAP functions that are not in MicroStrategy function object, such as RunningSlope.

Example:

```sql
ApplyRelative("RunningSlope(#0, #1)", [Metric 1], [Metric 2])
```

### ApplyComparison

ApplyComparison is used to define a filter. It can take facts, metrics, and attributes as input.

ApplyComparison can be used to define custom filters. In a SQL statement, it is used to populate the Where clause. It is the user’s responsibility to ensure the correctness of the expression entered inside the ApplyComparison function.

Examples

- ```sql
    ApplyComparison("#0 between #1 and #2", ?[Value Prompt Date], [Order Date]@ID, [Ship Date]@ID)
  ```
- ```sql
    ApplyComparison("#0>#1", Store@ID, Month@ID)
  ```
ApplyLogic

ApplyLogic is used to define a filter. The difference between ApplyLogic and ApplyComparison is that it takes logic (Boolean value), instead of value, as input.

Example

ApplyLogic("#0 AND #1", Year@ID > 2002, Month@ID > 200201)
Introduction

This appendix introduces some common issues whose solutions can add some complexity to the logical data model. While the topics are largely related to logical model design, a working knowledge of physical schemas will help when dealing with the challenges involved with these topics.

Before reading this appendix, you should know what logical data models and physical warehouse schemas are, and how to read and interpret them.
Attribute relationship

Building an effective project in MicroStrategy requires the project designer to have a solid understanding of all the attributes in the project, as well as how each of them relates to the other attributes.

Attributes are either related or nonrelated to each other:

- **Unrelated**—No parent-child relationship has been defined in Architect. No relationship is present in the lookup tables or relationship tables for these attributes. Unrelated attributes can exist together in fact tables, giving context to the fact.

  For example, the Customer and Date attributes have no relationship to one another. A particular customer and a particular date only make sense together if a fact is associated with that combination, for example, Amy spent $20 on January 5, 2003.

- **Related**—A parent-child relationship is defined in Architect between two or more attributes. In these cases, the relationship is defined through the attribute’s lookup table or a relationship table.

  For example, the Country and City attributes have a one-to-many relationship and are easily related through City's lookup table, which includes City_ID and Country_ID. In another example, the Item and Color attributes could have a many-to-many relationship and are related through a relationship table.

The implications of whether attributes are related become clear when you begin building reports. You can run a report with two attributes that are related—Country and City, for example—without problems. A report with two nonrelated attributes, however, must include a metric based on a fact that is on or below the level of the two attributes, or else a cartesian join will result.

Alternatively, you can apply an advanced set filter called a relationship filter to the report and force the query to use a specific table that establishes a relationship. Additional information on relationship filters may be found in Chapter 5, *Filters*. 
Many-to-many relationships

The presence of many-to-many relationships introduces complexity and additional considerations that you must make to ensure an effective warehouse design.

Below are some real-life examples of many-to-many relationships, which must be carefully handled in the data model and schema:

- In a certain organization, each salesperson can work in more than one calling center. Likewise, each calling center has many salespeople.
- In a car manufacturing plant, many models of cars are produced, and each comes in several colors. That is, there are many colors for a single type of car, and many types of cars can be associated with the same color.

The example in this appendix uses items and colors to demonstrate a many-to-many relationship and the options you have for dealing with them. In this case, one item can come in many colors—red hats, blue hats, green hats—and one color can be associated with many items—red dress, red hat, red shoes, red socks.

Potential problems with many-to-many relationships usually come in the following forms, both of which can be avoided by correctly modeling the relationship:

- loss of analytical capability
- multiple counting

Loss of analytical capability

With the color/item many-to-many relationship, there are usually two business questions for which users want answers:

1. In what colors are certain items available?
2. How much of a particular item/color combination was sold?
Answering the first question requires a table that contains a list of all possible item/color combinations. Recall that one-to-many relationships are usually in the child's lookup table.

In many-to-many relationships this is not feasible. Rather, a distinct relationship table needs to be present in your warehouse. The following diagram shows the lookup and relationship tables for item and color:

![Diagram showing item, color, lookup color, lookup item, and relationship table]

The Rel_Color_Item table provides a row for every possible item/color combination.

Answering the second question requires a fact table that has sales information as well as color and item information. The following diagram shows the same scenario as before, but in addition it shows a simple fact table containing sales data keyed by item, color, and date.

![Diagram showing item, color, lookup color, lookup item, relationship table, and fact table]
Note that this fact table alone is not sufficient to answer the first question. Only item/color combinations that were actually sold—and therefore have sales recorded—can be retrieved from this table. If you have item/color combinations that are available but that have never been sold, this fact table will not be able to provide a complete list of item/color combinations to answer question one.

In summary, to prevent any loss of analytical flexibility when dealing with many-to-many attribute relationship, you must always have two things:

- a distinct relationship table to identify all the possible combinations of attribute elements between attributes
- both the attribute ID columns in the fact table

There are several ways to implement the above points. These will be discussed later in this appendix.

**Multiple counting**

When dealing with many-to-many relationships, loss of analytical capability is only one challenge. Another equally significant issue is multiple counting. Multiple counting occurs when you attempt to aggregate data to the level one or higher attribute in the many-to-many relationship, and the relationship exists in a distinct relationship table but both attributes are not in the fact table. Recall the example from above, but make the following change: remove color from the fact table.
Assume that there are three items—hats, dresses, and socks—and that they come in three colors—red, blue, and green—with the exception of socks, which only come in green and blue. The following diagram shows this data in the lookup tables as well as some simple sales data:

The risk of multiple counting occurs when you run a query requesting the sales by color, effectively aggregating to the item attribute level in the many-to-many relationship. This query would require both the fact table—which has the sales information by item—and the relationship table—since color is not recorded in the fact table.
The difficulty lies in the fact that color is not in the fact table. There is no way to directly relate the sales of an item in the fact table to the color of that particular item. For example, instead of calculating the sales of red items, the query aggregates sales for all items that come in red according to the relationship table. The sum includes all hats and all dresses, including blue ones and green ones. This obviously leads to numbers that are higher than the true sales for red items.

For example, using the given data, the answers to the following questions is as follows:

- What are the total sales for hats?
  
The answer is $35, and this can be calculated directly from the fact table.

- What are the total sales for red items?
  
  There is no way to figure this out accurately. The answer you will get is $85, which is the total for all hats and dresses, since socks do not come in red. It is entirely possible that all the dresses sold are green; however, there is no way to know since color is not recorded in the fact table.

- What are the total sales for red dresses?
  
  Again, there is no way to know. If all the dresses sold are indeed green, the correct answer is $0, but the answer you will get based on the data in the fact table is $50.

Notice that this problem only occurs when you attempt to aggregate data to the level of one of the attributes. The following section describes several ways to prevent multiple counting when dealing with many-to-many relationships.
Working with many-to-many relationships

As you can probably already see, seemingly simple questions require you to take a number of steps to answer them.

There are three ways in which you can provide physical support to answer the types of questions you have seen so far. The three ways all have differing levels of flexibility, and flexibility is always a trade-off with complexity. In all cases, the two fundamental components remain in place in one form or another:

- a relationship table to define the attribute relationship
- both the attribute’s ID columns in the fact table

Remember, MicroStrategy Architect builds the rules that MicroStrategy SQL Engines uses to generate SQL when a report request is made. If you make both of the above physical implementations, the SQL Engine uses the related table when no metric is include on the report. When a metric is included, the fact table is used to answer the query.

All of the following methods require additional data in the fact table. This means that you must capture the additional data in the source system. That is, you need to have data in the source system as to what the color of each item was when it was sold. If this additional data was never captured in the source system, there is nothing you can do to resolve the many-to-many relationship.

Method 1

This method is the most straightforward way to effectively manage many-to-many relationships.

Method 1 requires you to create a separate relationship table and add both attribute IDs to the fact table as shown in the following diagram.
Method 2

Method 2 eliminates the many-to-many relationship and the need for a distinct relationship table.

Here the many-to-many relationship is converted into a compound attribute relationship. You treat one attribute as a child of the other and have a compound key for the lower level attribute. Also, you add both attribute IDs to the fact table as shown in the following diagram.

While this method eliminates the need for a separate relationship table, you lose the ability to view items independent of color, or vice versa.
Method 3

Method 3 is the most versatile solution and it has the following characteristics:

- further simplifies the compound attribute relationship from Method 2 into a simple attribute relationship
- provides the ability to view item and color together or independently
- requires only one attribute column in the fact table for complete flexibility, rather than two

Here you must create a new attribute, lower in level than either color or item. This attribute is essentially a concatenation of color and item, which gives it a one-to-many relationship between itself and each of its parent attributes. This is the SKU attribute, particularly common in retail data models or situations.

Finally, rather than including Color and Item in the fact table, you only need to include this new child attribute SKU, as shown in the following diagram.

This method is actually quite similar to Method 1. The major difference is that the distinct relationship table from Method 1 has an additional column, SKU, which extends the relationship of each item/color combination into a single value. Consequently, you can use this single value in the fact table.

The major disadvantage of Method 3 lies in creating the new attribute if your business model does not already use a similar structure and possibly adding complexity to the ETL process.
Joint child relationships

What is a joint child relationship?

Joint child relationships are special attributes that are sometimes called cross-dimensional attributes, text facts, or qualities. They do not fit neatly into the modeling schemes you have learned about thus far. These relationships can be modeled and conceptualized like traditional attributes, but like facts, they exist at the intersection of multiple attribute levels.

Many source systems refer to these special attributes as flags. So if you find flags in your source system documentation, these are likely candidates for joint child relationships.

Joint child relationships are really another type of many-to-many relationship where one attribute has a many-to-many relationship to two otherwise unrelated attributes.

For example, consider the relationship between three attributes: promotion, item, and quarter. In this case, promotion has a many-to-many relationship to both item and quarter as shown in the following diagram.

![Diagram showing a joint child relationship between promotion, quarter, and item]

Note: Quarter and Item have no direct relationship to one another.
An example of a promotion might be a “Red Sale” where all red items are on sale. A business might run this promotion around Valentine's Day and again at Christmas time.

**Supporting joint child relationships**

As you learned in the previous topic on many-to-many relationships, one way to resolve a many to many relationship is to have a relationship table for the attributes involved in the many-to-many relationships. In this case, you might create two relationship tables, one to relate promotion and item and another to relate promotion and quarter as shown in the following diagram.

These two tables are sufficient to answer questions like:

- What items have been in what promotions?
- What quarters have had what promotions?

However, these tables are not sufficient to answer the following more detailed and insightful questions:

- What items were in what promotions in a given quarter?
- In what quarters was a certain item involved in a certain type of promotion?
To answer these questions, you need to combine the two relationship tables, creating one table to relate all three attributes.

The relationship in the distinct relationship table must exist for a joint child relationship to be properly defined. However, it does not necessarily have to be in its own, distinct relationship table. Defining the relationship directly in the lookup table for the parent of the joint child—in this case, promotion—would be fine. Alternatively, you could build the relationship directly into the fact table.

The most important thing to notice about these examples is the relationship between the three attributes. The promotion attribute is really related to a particular item-quarter pair, as opposed to it being related to item and quarter separately. This is the essence of a joint child relationship and is shown in the following diagram.

Notice that a joint child relationship can be one-to-many or many-to-many. The issues with many-to-many relationships—loss of analytical capability and multiple counting—also apply to many-to-many joint child relationships.
If you have a joint child relationship in your data, it is important for you to define it in MicroStrategy Architect so that you get the correct data for reports that use the parent attribute in a joint child attribute. This ensures that when you need to join the fact table to the parent attribute of a joint child relationship—to see sales by promotion, for example—the join will always use both joint children rather than just one or the other.

### Attribute roles

**Attribute role** is a term used to define the use of a lookup table that is used for more than one attribute. For each attribute, the meaning is slightly different. In the following diagram, state is an example of a role since it relates to both the vendor and store attributes. In one case, it refers to the location of a vendor. In the other case, it refers to the location of a store. The state attribute is therefore said to be playing two roles.

In an OLTP system, roles are most often implemented as a single table, as shown in the above diagram. In the data warehouse, a query involving both vendor state and store state would need to use the State table twice in the same query. For example, a report is created to display vendors from Arkansas who sold to New York stores. The results may be blank if the data warehouse structure was set up incorrectly. The SQL statement tries to obtain the description of a state that is both Arkansas and New York simultaneously, generating the empty result set.
To see both roles on the same report, you must treat them as different attributes. That is, they must have different attribute names. To create unique attributes, you have two options:

- **automatic attribute role recognition**, where you create multiple attributes that have the same lookup table and allow MicroStrategy to automatically detect the multiple roles

- **explicit table aliasing**, where you create multiple logical tables pointing to the same physical table and define those two logical tables as the lookup tables for the two attributes

If you are creating new metadata in MicroStrategy 7.2, automatic recognition is turned on by default. If you are upgrading from previous versions, automatic recognition is turned off. Automatic recognition is a VLDB property on the project level.

Table aliasing provides advanced users with more control. If you are upgrading or have a very complex schema, it may be the better alternative. If you are new to MicroStrategy, it is easier to use automatic attribute role recognition. MicroStrategy recommends that you take advantage of automatic recognition if either of the following is true:

- You do not know the details of the modeling logic or the database.
- You are upgrading from a MicroStrategy version earlier than 7.2 and did not use database views to implement multiple roles for an attribute.

Automatic recognition does not work if the attributes are in the same hierarchy, meaning that a child attribute is shared. In the example given, the two state attributes do not have a common child attribute.

In summary, if you identify that any one of your attributes needs to play multiple roles, an attribute must be created in the logical model for each of the roles. Remember this rule to help you identify attribute roles: If you want to see the same attribute multiple times on one report, as ship month and order month, for example, the attribute has multiple roles. You can use either automatic attribute role recognition or explicit table aliasing to create the attribute roles.
**Automatic attribute role recognition**

In the data warehouse, a query involving both vendor state and store state would need to use the State table twice in the same query to get correct results. You can set up two attributes, Store State and Vendor State, both of which use the same lookup table. The SQL code will contain a self-join with the LU_State table. The logical model would look like the following:

Note that both roles for the State attribute are included in the logical model so that “State” can be considered from two different perspectives. Since the state in which a vendor resides and the state in which one of our stores is located are two different things, the logical model must reflect that. Automatic recognition allows these two attributes to access the same lookup table, using different attribute names for the same expression.

Automatic role recognition works only when the attributes use exactly the same expression.

Consider the following sample desired report.
In this case, the request is, “Show me total sales by Store State for all my vendors in Arkansas (Store State ID = 15).” The same lookup table, LU_State, can be used for both attributes, Store State and Vendor State, if attribute roles are used. Then the two attributes refer to the same columns of that table.

To use automatic attribute role recognition, you must select the Engine Attribute Role Options, found in the Project Level VLDB Properties under Query Optimization.

**Explicit table aliasing**

Explicit table aliasing provides more robust functionality than automatic recognition, so advanced users are encouraged to take advantage of this solution.

To continue with the state example, the logical model would remain the same as with automatic recognition. Both roles for the State attribute are included in the logical model so that State can be considered from two different perspectives. Separate lookup tables are created in the schema, but they point to the same physical table. One table (LU_State_Store) contains the attribute Store State while the other (LU_State_Vendor) contains Vendor State.
Recall the sample report that displays total sales by Store State for all vendors in Arkansas. When explicit table aliasing is used, the two lookup tables LU_State_Store and LU_State_Vendor are used. Since they are just different names for the same physical table, the report actually accesses the same physical table, LU_State, for both state names, as shown by this sample SQL:

```
SELECT a12.State_Desc as State_Desc
SELECT a13.State_Desc as State_Desc
FROM LU_State a12
     LU_State a13
```

To create a table alias

1. On the MicroStrategy Desktop, navigate to the Tables folder, under the Schema Objects folder.

2. Right-click the table to alias and select Create Table Alias. This option makes a copy of table in the schema.

3. Type the table alias, or the name for the new table.

4. When creating the new attributes, select the appropriate table for each attribute. For example, in the case discussed above, you would select the LU_State_Store table for the Store State attribute and LU_State_Vendor for Vendor State.

See the online help for steps for creating attributes.
Logical and Mathematical Operators for Filtering

Introduction

As discussed in Chapter 5, Filters, you can use logical operators, such as AND, OR, NOT, and so on, to add additional qualifications to filters and report limits. These logical operators set conditions for the report query to retrieve data from the data warehouse and display the data in the report.

This appendix discusses operators, specifically what logical operators are and how to use them.
What is an operator?

*Operators* are used to manipulate individual data items and data sets. These data items are called operands or arguments. Operators are represented by special characters or by keywords. For example, multiplication is represented by an asterisk (*) and division is represented by a slash (/). Filtering conditions are expressions built from attribute forms, metrics, constants, expressions, and operators. For example, consider the following filtering definition:

\[ \text{Store\_ID} = 1 \]

The definition above contains an attribute (Store), an attribute form (Store\_ID), a comparison operator (=), and a numeric constant (1).

The following types of operators are used when specifying filtering conditions:

- **logical**
- **comparison**
- **rank and percent**
- **pattern**

Logical operators

*Logical operators* allow the application of certain conditions to two sets of filter expressions simultaneously. There are three basic logical operators:

- Union behaves as the inclusive term **OR** does in grammar. The union of two sets yields a TRUE value any time that either or both of the sets of filtering criteria are met.

- Intersection behaves as the term **AND** does in grammar. The intersection of two sets yields a TRUE value only when both sets of filtering criteria are met.
• Exclusion behaves as the term **AND NOT** does in grammar. When two sets of filtering criteria are linked in this manner, their combination yields a TRUE value only when the first set is met, and the other set is not satisfied.

The following tables show the combinations possible with each logical operator, and the value that each combination yields, using the following filtering criteria as an example:

\[
\begin{align*}
A &= \text{(customers located in the) Northeast region} \\
B &= \text{(customers that purchased) blankets}
\end{align*}
\]

### Logical union filter: A OR B

Possible filter combinations resulting from the union of attributes A and B (customers that either are located in the Northeast region **OR** have purchased blankets) are as follows.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Result Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TRUE</td>
<td>customers located in the Northeast region OR customers that purchased blankets</td>
</tr>
<tr>
<td>2</td>
<td>FALSE</td>
<td>customers that purchased blankets (but are not located in the Northeast region)</td>
</tr>
<tr>
<td>3</td>
<td>TRUE</td>
<td>customers located in the Northeast region (but have not purchased blankets)</td>
</tr>
<tr>
<td>4</td>
<td>FALSE</td>
<td>no display (customers that are neither located in the Northeast region nor purchased blankets)</td>
</tr>
</tbody>
</table>

Because a union of two sets yields a valid result if data corresponding to either set is found, this filter causes a display as shown in rows 1, 2, and 3.
Logical intersection filter: A AND B

Possible filter combinations resulting from the intersection of attributes A and B (customers that are located in the Northeast region AND have purchased blankets) are as follows.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Result Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>Customers that are located in the Northeast region AND have purchased blankets</td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td>No display (customers that purchased blankets but are not located in the Northeast region)</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td>No display (customers that are located in the Northeast region but have not purchased blankets)</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td>No display (customers that neither are located in the Northeast region nor have purchased blankets)</td>
</tr>
</tbody>
</table>

Because an intersection of two sets yields a valid result only if data corresponding to both sets is found, this filter causes display as shown in row 1, and in no other combination.

Logical exclusion filter: A AND NOT B

Possible filter combinations resulting from the not (and not) exclusion of an attribute (for example, B) (customers that are located in the Northeast region AND have not purchased blankets) are as follows.

<table>
<thead>
<tr>
<th>A</th>
<th>NOT B</th>
<th>Result Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>Customers who are located in the Northeast region AND have not purchased blankets</td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td>No display (customers who have not purchased blankets AND are not located in the Northeast region)</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td>No display (customers that are located in the Northeast region AND have purchased blankets)</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td>No display (customers not located in the Northeast region who have purchased blankets)</td>
</tr>
</tbody>
</table>
The behavior of exclusive *not (and not)* statements is the same as that of intersections — the combination yields a valid result only when data corresponding to the “included” set is found and data corresponding to the “excluded” set is not. This filter causes display as shown in row 1 and in no other combination.

**Logical exclusion filter: A OR NOT B**

Possible filter combinations resulting from the + not (or not) exclusion of an attribute (for example, B) (customers that are located in the Northeast region OR have not purchased blankets) are as follows.

<table>
<thead>
<tr>
<th></th>
<th>Result Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Customers who are located in the Northeast region OR have not purchased blankets</td>
</tr>
<tr>
<td>2</td>
<td>Customers who have not purchased blankets OR are not located in the Northeast region</td>
</tr>
<tr>
<td>3</td>
<td>Customers that are located in the Northeast region OR have purchased blankets</td>
</tr>
<tr>
<td>4</td>
<td>No display (customers not located in the Northeast region who have purchased blankets)</td>
</tr>
</tbody>
</table>

The behavior of exclusive *or not* statements is the same as that of unions—the combination yields a valid result when either data corresponding to the “included” set is found or data corresponding to the “excluded” set is not. This filter causes display as shown in rows 1, 2, and 3.

**Comparison operators**

Comparison operators compare values. The values can be numbers, text strings, or expressions. The comparison operators are as follows:

- Between: Identifies values in a range that includes both a lower and an upper limit. For example, “between 10 and 20” returns all values that are greater than or equal to 10 and less than or equal to 20.
• Different from: Identifies values that are other than the specific value indicated. For example, “different from 10” returns all values that are not 10.

• Exactly: Identifies an specific value. For example, “exactly 1” returns all items with a value of 1.

• Greater than: Identifies values that are greater than an indicated lower limit. For example, “greater than 10” returns values that are greater than 10.

• Greater than or equal to: Identifies values that are greater than or equal to the limit indicated. For example, “greater than or equal to 10” returns all values that are 10 or greater.

• Less than: Identifies values that are less than an indicated upper limit. For example, “less than 10” returns values that are less that 10.

• Less than or equal to: Identifies values that are less than or equal to the limit indicated. For example, “less than or equal to 10” returns all values that are 10 or less.

• Not between: Identifies values that are outside a specified range. For example, “not between 10 and 20” returns all values that are less than 10 and greater than 20.

• Is null: Identifies values that are null.

• Is Not null: Identifies values that are not null.

When you use these operators on a description for a date, the results can appear to be incorrect. For example, assume the date description is formatted as Jan 2003. Create a filter on the description attribute using the Between operator to return the months between January 2003 and June 2003. The results are Jan 2003, Jul 2003, and Jul 2003, not the first six months of the year as expected. This occurs because the description attribute is formatted as text, not as numbers or dates, and therefore the SQL sorts the data alphabetically. To obtain results of January 2003, February 2003, March 2003, April 2003, May 2003, and June 2003, filter on the ID rather than the description or use the in list operator.
Rank and percent operators

Rank and percent operators are only applicable to metrics. The following operators are visible when you qualify on the Rank or Percent function.

- **Between**: Identifies values in a range that has both a lower and an upper limit. For example, “between 10 and 20” returns all values that are greater than or equal to 10 and less than or equal to 20.

- **Bottom**: Identifies the lowest set of values in a range. For example, “bottom 40” returns the 40 lowest values within a given range selected.

  In MicroStrategy Web, this is called Lowest.

- **Different from**: Identifies values that are other than the specific value indicated. For example, “different from 10” returns all values that are not 10.

- **Exactly**: Identifies a specific value. For example, “exactly 1” returns all items with a value of 1.

- **Exclude top**: Is used in rank and percent calculations, discards a range of top values from a given set. For example, “exclude top 10” returns all values in the set but the top 10 percent.

- **Exclude bottom**: Is used in rank and percent calculations, discards a range of bottom values from a given set. For example, “exclude bottom 10” returns all values in the set but the bottom 10 percent.

- **Is null**: Identifies values that are null

- **Is not null**: Identifies values that are not null

- **Not between**: Identifies values that are outside a specified range. For example, “not between 10 and 20” returns all values that are less than 10 and greater than 20.

- **Top**: Indicates the topmost value range in a given set. For example, “top 40” returns the 40 highest values in a set.

  In Web, this is called Highest.
Pattern operators

Pattern operators allow text strings to be compared. Pattern operators are case-sensitive. The following pattern operators are available in MicroStrategy:

- Begins with: Returns a result set that starts with a specified value. For example, “begins with J” returns all strings beginning with J, that is, January, June, and July.

- Ends with: Returns a result set that ends with a specified value. For example, “end with r” returns all strings ending with r, that is, September, October, and all the other months meeting the criterion.

- Contains: Returns a result set that contains a specified value. For example, “contains ua” returns all strings that contain ua, such as January and February.

- Does not begin with: Returns a result set that does not start with a specified value. For example, “does not begin with J” returns only those values that do not begin with J, such as May, February, October, and so on.

- Does not end with: Returns a result set that does not end with a specified value. For example, “does not end with r” returns only those strings that do not end with r, such as March, April, and all the other months meeting the criterion.

- Does not contain: Returns a result set that does not contain a specified value. For example, “does not contain ua” returns only those values that do not contain ua, such as March, May, and so on.
Introduction

In all supported warehouse platforms other than Microsoft Access, MicroStrategy uses SQL statements to query the relational database management system (RDBMS) catalog tables to obtain warehouse catalog information. This includes catalog tables, columns, and their data types. These Catalog SQL statements vary from platform to platform and can be customized according to the characteristics of the specific warehouse.

Microsoft Access does not have catalog tables, so an ODBC call must be used to retrieve information about tables and columns in Access. By default, a similar ODBC call is used for the Generic DBMS database type, but you can choose to use custom catalog SQL for the generic type if you wish.

This appendix discusses customizing SQL statements, the structure of the SQL Catalogs, and the default SQL statements used for each data warehouse.
Customizing Catalog SQL statements

The MicroStrategy Warehouse Catalog can be configured to read the catalog information in one- or two-pass SQL mode. In two-pass SQL mode, it first reads only the tables from the warehouse. The structure of individual tables is read only when the table is selected. This is the recommended option for interactive warehouse catalog building because no unnecessary catalog information is read from the warehouse, which increases processing speed. One-pass SQL mode, on the other hand, reads all the tables and columns in one SQL statement. This option is recommended only if the Catalog SQL is well customized to limit the amount of data returned by it.

The two retrieval options use different Catalog SQL, but both can be customized in the Warehouse Catalog Options dialog. In the following sections, the name Catalog Table SQL refers to the Catalog SQL to retrieve the tables in the warehouse; that is, the first SQL used in a two-pass catalog retrieval. The name Full Catalog SQL refers to the SQL used to read all the tables and columns in one pass. To customize a Catalog SQL, you must understand several important concepts: the table name space, SQL template strings, and incomplete Catalog SQL.

The table name space

In a typical RDBMS platform, a table name does not uniquely identify it in a particular warehouse database installation. A table name space is a partition of the warehouse installation in which table names are unique. Depending on the type of RDBMS, this name space can be the name of the warehouse database, the owner of the table, or a combination of both database and owner. In both the Catalog Table SQL and Full Catalog SQL, a name space gives each table a unique name. This helps you to avoid confusing tables that share the same table name.

The table name space is optional. A customized Catalog SQL can omit the name space if duplicate table names do not present a problem in the warehouse database.
SQL template strings and incomplete Catalog SQL

The default system Catalog SQL can contain certain template strings that can be resolved at run time or must be completed manually by the user. These templates are listed here:

- **#LOGIN_NAME#**—This template is automatically replaced at run time with the login name used to connect to the warehouse. You can leave this template in the customized SQL if you want the Catalog SQL to yield different results depending on the warehouse login used. Otherwise, this template is replaced with the name of the warehouse user who owns the warehouse tables of interest.

- **#?Database_Name?, #?Schema_Name?#**—This Catalog SQL template is an incomplete SQL string that must be completed by the user before it can be executed. The string starts with “#?” and ends with “?#”. The command #?Database_Name?, used with Teradata, must be replaced with the name of the warehouse database containing the warehouse tables. #?Schema_Name?, used with DB2 AS/400, must be replaced with the name of the schema in which the warehouse tables reside.

Structure of Catalog Table SQL

Catalog Table SQL is expected to return two columns, one identifying the name space of the table and the other the name of the table. If a name space is not provided, only the table name column is required. Each row of the SQL result must uniquely identify a table. Duplicates are not allowed. The column that identifies the table name space uses the SQL column alias NAME_SPACE. The column that identifies the table name has the alias TAB_NAME. The following example is the default Catalog Table SQL for Oracle 8.0:

```sql
SELECT DISTINCT OWNER NAME_SPACE,
   TABLE_NAME TAB_NAME
FROM ALL_TAB_COLUMNS
WHERE OWNER = ' #LOGIN_NAME# '"
```
Structure of Full Catalog SQL

Full Catalog SQL is expected to return between five and seven columns, depending on the RDBMS platform and the customization. The following aliases are required to identify each column returned:

- **NAME_SPACE** (optional)—the table name space
- **TAB_NAME** (required)—name of the table
- **COL_NAME** (required)—name of the column
- **DATA_TYPE** (required)—a string or a number that identifies the major datatype of the column
- **DATA_LEN** (required)—a number that describes the length or size of the column data
- **DATA_PREC** (optional)—a number that describes the precision of the column data
- **DATA_SCALE** (optional)—a number that describes the scale of a floating point column data

Full Catalog SQL must return its rows ordered first by **NAME_SPACE**, if available, and then by **TAB_NAME**.

The following example is the default Full Catalog SQL for Microsoft SQL Server 7.0:

```sql
SELECT U.name NAME_SPACE, T.name TAB_NAME, 
    C.name COL_NAME, C.type DATA_TYPE, 
    C.length DATA_LEN, C.prec DATA_PREC, 
    C.scale DATA_SCALE
FROM sysobjects T, syscolumns C, sysusers 
WHERE T.id = C.id and T.type in ('U', 'V') 
    AND T.uid = U.uid 
ORDER BY 1, 2
```
Default Warehouse Catalog SQL

The following table shows the default Warehouse Catalog SQL used by MicroStrategy for each supported warehouse platform. You are encouraged to consult this table before writing your own customized Catalog SQL.

<table>
<thead>
<tr>
<th>RDBMS</th>
<th>Default Catalog Table SQL</th>
<th>Full Catalog SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM DB2 OS/390</td>
<td><code>SELECT TBCREATOR NAME_SPACE, TBNAME TAB_NAME</code>&lt;br&gt;<code>FROM SYSIBM.SYSCOLUMNS</code>&lt;br&gt;<code>WHERE TBCREATOR='#LOGIN_NAME#'</code></td>
<td><code>SELECT TBCREATOR NAME_SPACE, TBNAME TAB_NAME, NAME COL_NAME, COLTYPE DATA_TYPE, LENGTH DATA_LEN, SCALE DATA_SCALE</code>&lt;br&gt;<code>FROM SYSIBM.SYSCOLUMNS</code>&lt;br&gt;<code>WHERE TBCREATOR='#LOGIN_NAME#'</code>&lt;br&gt;ORDER BY 1, 2</td>
</tr>
<tr>
<td>IBM DB2 AS/400</td>
<td><code>SELECT DISTINCT SYSTEM_TABLE_SCHEMA NAME_SPACE, TABLE_NAME TAB_NAME</code>&lt;br&gt;<code>FROM QSYS2.SYSCOLUMNS</code>&lt;br&gt;<code>WHERE TABLE_OWNER = '#LOGIN_NAME#' AND SYSTEM_TABLE_SCHEMA = '#?Schema_Name?#'</code></td>
<td><code>SELECT SYSTEM_TABLE_SCHEMA NAME_SPACE, TABLE_NAME TAB_NAME, COLUMN_NAME COL_NAME, DATA_TYPE DATA_TYPE, LENGTH DATA_LEN, NUMERIC_SCALE DATA_SCALE</code>&lt;br&gt;<code>FROM QSYS2.SYSCOLUMNS</code>&lt;br&gt;<code>WHERE TABLE_OWNER = '#LOGIN_NAME#' AND SYSTEM_TABLE_SCHEMA = '#?Schema_Name?#'</code>&lt;br&gt;ORDER BY 1, 2</td>
</tr>
<tr>
<td>IBM DB2 UDB</td>
<td><code>SELECT DISTINCT TBCREATOR NAME_SPACE, TBNAME TAB_NAME</code>&lt;br&gt;<code>FROM SYSIBM.SYSCOLUMNS</code>&lt;br&gt;<code>WHERE TBCREATOR = '#LOGIN_NAME#'</code></td>
<td><code>SELECT TBCREATOR NAME_SPACE, TBNAME TAB_NAME, NAME COL_NAME, COLTYPE DATA_TYPE, LENGTH DATA_LEN, SCALE DATA_SCALE</code>&lt;br&gt;<code>FROM SYSIBM.SYSCOLUMNS</code>&lt;br&gt;<code>WHERE TBCREATOR='#LOGIN_NAME#'</code>&lt;br&gt;ORDER BY 1, 2</td>
</tr>
</tbody>
</table>

*Note: requires manual replacement of template string #?Schema Name?#*
<table>
<thead>
<tr>
<th>RDBMS</th>
<th>Default Catalog Table SQL</th>
<th>Full Catalog SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic DBMS</td>
<td>* Note: By default, SQL is not used to query the catalog for this RDBMS type. If you do wish to use SQL, it must conform to the structure as outlined in the SQL template shown here.</td>
<td>SELECT Name Space NAME_SPACE, Table Name TAB_NAME, Column Name COL_NAME, Data Type DATA_TYPE, Data Length DATA_LEN, Data Scale DATA_SCALE FROM ... WHERE TBNAME in (#TABLE_LIST#) ORDER BY 1, 2</td>
</tr>
<tr>
<td>Informix 7.x, 8.x, 9.x</td>
<td>SELECT DISTINCT owner NAME_SPACE, tabname TAB_NAME FROM SYSTABLES WHERE tabid &gt;= 100 AND tabtype IN ('T', 'V')</td>
<td>SELECT T.owner NAME_SPACE, T.tabname TAB_NAME, C.colname COL_NAME, C.coltype DATA_TYPE, C.collength DATA_LEN FROM SYSTABLES T, SYSCOLUMNS C WHERE T.tabid = C.tabid AND T.tabtype IN ('T', 'V', 'S') ORDER BY 1, 2</td>
</tr>
<tr>
<td>Oracle 7.3.x, 8.0.x</td>
<td>SELECT DISTINCT OWNER NAME_SPACE, TABLE_NAME TAB_NAME FROM ALL_TAB_COLUMNS WHERE OWNER = '#LOGIN_NAME#'</td>
<td>SELECT OWNER NAME_SPACE, TABLE_NAME TAB_NAME, COLUMN_NAME COL_NAME, DATA_TYPE TYPE DATA_TYPE, DATA_LENGTH DATA_LEN, DATA_PRECISION DATAPREC, DATA_SCALE DATA_SCALE FROM ALL_TAB_COLUMNS WHERE OWNER = '#LOGIN_NAME#' ORDER BY 1, 2</td>
</tr>
<tr>
<td>Oracle 8i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Red Brick 5.x, 6.x</td>
<td>SELECT DISTINCT CREATOR NAME_SPACE, NAME TAB_NAME FROM RBW_TABLES WHERE ID &gt; 0 AND CREATOR='#LOGIN_NAME#'</td>
<td>SELECT T.CREATOR NAME_SPACE, T.NAME TAB_NAME, C.NAME COL_NAME, C.TYPE DATA_TYPE, C.LENGTH DATA_LEN, C.PRECISION DATAPREC, C.SCALE DATA_SCALE FROM RBW_TABLES T, RBW_COLUMNS C WHERE T.ID = C.TID AND T.ID &gt; 0 ORDER BY 1, 2</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Default Catalog Table SQL</td>
<td>Full Catalog SQL</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Microsoft SQL Server 7.0</td>
<td>SELECT DISTINCT U.name NAME_SPACE, T.name TAB_NAME FROM sysobjects T, sysusers U WHERE T.uid = U.uid AND T.type IN ('U', 'V')</td>
<td>SELECT U.name NAME_SPACE, T.name TAB_NAME, C.name COL_NAME, C.type DATA_TYPE, C.length DATA_LEN, C.prec DATA_PREC, C.scale DATA_SCALE FROM sysobjects T, syscolumns C, sysusers U WHERE T.id = C.id and T.type in ('U', 'V') AND T.uid = U.uid ORDER BY 1, 2</td>
</tr>
<tr>
<td>Sybase Adaptive Server 11.x, 12.x</td>
<td>SELECT DISTINCT U.name NAME_SPACE, T.name TAB_NAME FROM sysobjects T, sysusers U WHERE T.uid = U.uid AND T.type IN ('U', 'V')</td>
<td>SELECT U.name NAME_SPACE, T.name TAB_NAME, C.name COL_NAME, C.type DATA_TYPE, C.length DATA_LEN, C.prec DATA_PREC, C.scale DATA_SCALE FROM sysobjects T, syscolumns C, sysusers U WHERE T.id = C.id and T.type in ('U', 'V') AND T.uid = U.uid ORDER BY 1, 2</td>
</tr>
<tr>
<td>Sybase IQ 12</td>
<td>SELECT DISTINCT U.name NAME_SPACE, T.table_name TAB_NAME FROM systable T, sysusers U WHERE T.creator = U.uid AND T.table_type IN ('BASE', 'VIEW')</td>
<td>SELECT U.name NAME_SPACE, T.table_name TAB_NAME, C.cname COL_NAME, C.coltype DATA_TYPE, C.length DATA_LEN, C.syslength DATA_SCALE FROM systable T, syscolumns C, sysusers U WHERE T.table_name = C.tname and T.table_type in ('BASE', 'VIEW') AND T.creator = U.uid ORDER BY 1, 2</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Default Catalog Table SQL</td>
<td>Full Catalog SQL</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Tandem NonStop SQL</td>
<td>SELECT DISTINCT U.name NAME_SPACE, T.name TAB_NAME FROM sysobjects T, sysusers U WHERE T.uid = U.uid AND T.type IN ('U', 'V')</td>
<td>SELECT U.name NAME_SPACE, T.name TAB_NAME, C.name COL_NAME, C.type DATA_TYPE, C.length DATA_LEN FROM sysobjects T, syscolumns C, sysusers U WHERE T.id = C.id and T.type in ('U', 'V') AND T.uid = U.uid ORDER BY 1, 2</td>
</tr>
<tr>
<td>NCR Teradata</td>
<td>SELECT DISTINCT DatabaseName NAME_SPACE, TableName TAB_NAME FROM DBC.TABLES WHERE DatabaseName = '#?DATABASE_NAME?#'</td>
<td>SELECT DatabaseName NAME_SPACE, TableName TAB_NAME COL_NAME, ColumnType DATA_TYPE, ColumnLength DATA_LEN, DecimalTotalDigits DATA_PREC, DecimalFractionalDigits DATA_SCALE FROM DBC.COLUMNS WHERE DatabaseName = '#?DATABASE_NAME?#' ORDER BY 1, 2</td>
</tr>
</tbody>
</table>

*Note: requires manual replacement of template string #?Database_Name?#*
**Introduction**

You should already know how to create a simple project using Project Builder, which is covered in the *Installation and Configuration Guide*. Project Builder is a streamlined tool to get you started quickly with a simple project. It contains only a small subset of MicroStrategy’s functionality, allowing you to rapidly create user hierarchies and simple metrics and reports. You can use Project Builder to efficiently create quick proof-of-concept test projects with your own data. It is best suited for a basic setup procedure, resulting in a simple yet completely functional project.

While the Project Creation Assistant performs the same basic operation—that is, creating a project—its audience is experienced project creators who have planned all their facts, attributes, and relationships. The advantage of the Project Creation Assistant is its ability to create many schema objects at once. Since you can efficiently add many tables and develop numerous attributes and facts, it is especially useful for large projects which contain many tables and schema
objects. You can also create attributes with many-to-many relationships. Unlike Project Builder, which is targeted at new users and a basic setup, you cannot create metrics or reports with the Project Creation Assistant. Instead, you can use the Metric Editor and the Report Editor to create sophisticated and complicated metrics and reports.

Before you begin

Plan your project

You should plan your project before using the Project Creation Assistant. You should consider the following:

- the tables to use in the project
- the datatypes to use to identify facts
- the facts to include in the project
- the datatypes to use to identify attributes
- the attributes to create
- the description column name for each attribute
- any other attribute forms for each attribute

The additional attribute forms are not created through the Project Creation Assistant; you must add them through the Attribute Editor after you complete the Project Creation Assistant steps.

- the child attributes for each attribute
Create the metadata database

Before you access the Project Creation Assistant to generate a project, you should create the metadata database. The Configuration Wizard helps you create the empty metadata shell in the metadata database. These empty metadata tables will later be populated during the project creation step of the Project Creation Assistant. For more information on the Configuration Wizard, see the *Installation and Configuration Guide*.

You can also create a direct, or two-tier, project source to connect to the metadata database. However, setting up the project source within the Project Creation Assistant is easier. For further details on project sources, refer to the *Installation and Configuration Guide*.

Project creation

Now that you have planned your project and completed the prerequisites, you can use the Project Creation Assistant to guide you through building the project and populating the metadata based on the data structures present in the data warehouse.

The steps of the Project Creation Assistant are:

1. Initialize/create the project.
2. Select tables from the Warehouse Catalog.
3. Create facts.
4. Create attributes.

You should complete all the steps in the Project Creation Assistant at the same time. While you can save an incomplete project definition, you cannot finish creating it with the Project Creation Assistant. Instead, you must finish completing it using the appropriate interface, such as the Warehouse Catalog, Fact Creation Assistant, or Attribute Creation Assistant.
To access the Project Creation Assistant

- From the Schema menu, choose Create New Project. The Project Creation Assistant dialog box opens.

Initialize/create the project

The first step in project creation is to initialize the project using the New Project dialog box, which opens when you choose the Create project step from the Project Creation Assistant menu.

Initializing the project means identifying the new project with a name and the metadata repository in which to create the project—that is, the project source. If you have not already set up a project source, you can create one in the Project Source Manager by clicking New in the Project Source section.

To support anonymous authentication mode for guest users for this project, select Enable guest user account.

After you specify these initial settings, the shell of a project is created in the metadata. This means that the folder structure and default connectivity settings are set up. This process can take some time to complete.

If you are not authorized to create projects in the selected data source, the project will not be created.

Select tables from the Warehouse Catalog

This step in building a project determines the set of warehouse tables to be used, and therefore the set of data available to be analyzed in the project. The Warehouse Catalog is accessed to allow you to select the database tables to use in your new project. MicroStrategy schema objects such as attributes, facts, and tables are abstractions built on top of tables and columns in the database.
When you choose the Select tables step from the Project Creation Assistant menu, the Warehouse Database Instance dialog box opens. The database instance selected in this dialog box determines which data warehouse is accessed. If you did not previously set up a database instance, click New to create one in the Database Instance Wizard. You can also edit an existing database instance with the Database Instances dialog box. To access it, click Edit.

Once you set the database instance, the Warehouse Catalog dialog box opens. This screen lists all the tables in the database to which you are connected through your database instance and to which your database login has read privileges. You select the lookup, fact, and relationship tables to use in your new project. You should also include all the tables needed to complete your project, including transformation tables, aggregate tables, and partition mapping tables.

You can add additional tables after exiting the Project Creation Assistant by using the Warehouse Catalog. For more information, see the Installation and Configuration Guide.

To control how the warehouse tables are accessed and processed, options are offered in three different categories, catalog, view, and schema. To access them, click Options in the toolbar of the Warehouse Catalog dialog box.

**Catalog**

The Catalog options allow you to

- change the **database instance** to connect to a different data warehouse
- customize the **Catalog Read Method**, which affects how the tables and columns are retrieved from the database system catalog, with the following options:

  - **Settings** allows you to directly edit the SQL statements that are used to retrieve the list of available tables from the Warehouse Catalog and the columns for the selected tables. The default SQL retrieves a DISTINCT list of tables and columns from all users. You could constrict the information returned, for
example, by specifying certain conditions and/or table owners. For more information, see Appendix F, *Warehouse Catalog SQL*.

– You can choose whether to have the Project Creation Assistant **count the number of rows** for all tables when reading the database catalog.

– The final option is whether to **ignore the current table name space** when reading from the database catalog and update using new table name space.

- Switch the **catalog read method** from automatic to manual, so that the database is queried only after Read Catalog is selected. This allows you to control exactly how often the Warehouse Catalog is read for performance reasons. If you have a large database, the query can take some time. By default, the method is set to automatic to populate the table list when the Warehouse Catalog is opened.

Creating projects that use single-byte interfaces creates autostyles whose fonts are not compatible with double-byte characters. For example, you create a project in Desktop using the English interface, which is a single-byte language. If you then use a Chinese interface, which is a double-byte language, you can see Chinese characters on the Desktop. But if you export a report to PDF, the PDF uses autostyle fonts that are in English and loses the Chinese characters.

**View**

The View options allow you to

- set the **table prefix specifications**, which determine whether to display table prefixes in the table list. You can also attach a prefix to all the tables added to the project.

- set whether to **display the number of rows** for each table

- set whether to **display the name space** of each table
**Schema**

The Schema options allow you to

- Select whether **existing schema objects are automatically mapped** to new tables that are added to the project. The default is yes.

- Select whether to **automatically calculate the logical sizes** for any new tables added to the project. The default is yes.

After you select the tables to be added to the project, the table definitions are written to the metadata. This process can take some time to complete.

**Create facts**

This step accesses the Fact Creation Wizard to help you create the facts for your project. Facts relate numeric data values from the data warehouse to the MicroStrategy reporting environment. They allow you to access the data stored in the data warehouse and they form the basis for metrics.

**Fact creation rules**

The Fact Creation Wizard uses rules to help automate the fact creation process. These rules are accessed from the Define Rules button on the Introduction page of the wizard. The first rule determines, by datatype, which columns are displayed when you are selecting columns to use in facts. The second rule concerns how to create default fact names—whether to replace underscores in the fact name with spaces and whether the first letter is capitalized. You need to change these rules if the naming conventions in your warehouse do not conform to these defaults.
Fact column selection

Select the columns to be used as facts. You can rename any column to make it more user-friendly by selecting it and pressing F2.

The wizard cannot handle heterogeneous column names. Select each fact object only once. You can use the Fact Editor to add additional expressions, as well as fact extensions, after you complete the Project Creation Wizard. For more information on heterogeneous column names, see Chapter 10, Facts.

The selected fact definitions are written to the metadata.

Create attributes

This step accesses the Attribute Creation Wizard to help you create the attributes for your project. Attributes are used to group or aggregate fact data. An attribute acts like a column header, and the data that appears in the lookup table are elements. Elements define and make up the attribute.

Attribute creation rules

The Attribute Creation Wizard uses the rules listed below to help automate the attribute creation process. Change these rules if the naming or datatype conventions in your warehouse do not conform to these defaults. These rules are accessed from the Define Rules button on the Introduction page of the wizard.

- The column datatype rule determines, by datatype, which columns are available to be attribute ID columns.
- The attribute name rule concerns how to create default attribute names—whether to replace underscores in the attribute name with spaces, remove the word ID from the name, and capitalize the first letter.
• The warehouse search rule sets naming conventions to help locate your warehouse objects. The defaults are ID for identifier columns, DESC for description columns, and LOOKUP for lookup tables.

**ID column selection**

Select the columns to be used as attributes. Only those columns with datatypes that match those chosen in the rules appear here. To assist you, the columns that match the identifier naming convention set in the warehouse search rule are automatically highlighted. You can rename any attribute to make it more user-friendly by selecting it and pressing F2.

Ensure that all values in the ID column are unique and that it does not contain null values. Although Desktop allows it, a column that has null or repeated values should never be used as the ID column for an attribute. Unexpected behavior and errors result.

Compound attributes can also be created in this step. A compound attribute is an attribute where more than one ID column is needed to uniquely identify the elements of that attribute. For more information on compound attributes, see Chapter 11, *Attributes*.

**Description column selection**

For each attribute, you can select whether to use the ID or a different column for the description of the attribute. To help you, the column that follows the description naming convention set in the warehouse search rule is automatically selected.

Other attribute forms need to be created through the Attribute Editor after you complete steps in the Project Creation Assistant. Refer to Chapter 11, *Attributes*, for more information about attribute forms.
Lookup table selection

Select the lookup table for each attribute. Lookup tables are the physical representation of attributes; they provide the information for an attribute through data stored in their ID and description columns. To help you, the tables that follow the lookup naming convention set in the warehouse search rule is automatically selected.

Compound attribute definition

If you created compound attributes, their descriptions and lookup tables are selected separately from other attributes.

Relationship definition

For each attribute, you specify the children and the type of relationship: one-to-one, one-to-many, or many-to-many.

After you have completed these pages, the attributes are created.

Project completion

You have now completed making a project with the Project Creation Assistant. You can continue to develop the project, using the editors as described in the next section, to add complexity and flexibility to the project.

Additional schema configurations

You can configure additional schema-level settings to increase the flexibility of the project you produced with the Project Creation Assistant. These settings include

- Attribute definitions—The Attribute Creation Wizard allows you to quickly develop multiple attributes at once. You can use the Attribute Editor to define or edit more complex attributes.
- Fact definitions—The Fact Editor allows you to design derived and implicit attributes, heterogeneous mappings of warehouse columns, fact extensions, and fact degradations.

- User hierarchies—The Hierarchy Editor allows you to create user hierarchies, which facilitate access to attribute and element browsing and drilling.

- Advanced configurations—These objects include aggregate tables, partitioning and partition mappings, and transformations. The tools used to create them are the Warehouse Catalog, the Metadata Partition Mapping Editor, the Warehouse Partition Mapping Editor, and the Transformation Editor.

All these schema-level objects are discussed in this manual.
ETL INFORMATION

Description

The extraction, transformation, and loading (ETL) process represents all of the steps necessary to move data from disparate source systems to an integrated data warehouse.

The data is first extracted, or retrieved, from the source systems. It is then transformed before being loaded into the data warehouse. Transformation procedures can include converting datatypes and column names, eliminating bad data, correcting typographical errors, filling in incomplete data, and so on. The third and final step is to load the data into the warehouse.
ETL information is available from the ETL Information option on the Grid menu of the Report Editor. The data provided allows you to ascertain the origin and structure of the OLTP database table columns used to create columns in the warehouse tables from which attribute and metric data for the report has been obtained.

The option, which is available from the ETL Information option on the Grid menu, displays the following information for a selected attribute or metric on a report:

- a definition of the attribute
- a list of the tables used to define attribute elements or metric facts
- a list of the columns included in attribute or metric definition
- a list of names for the mappings used in transforming the columns from the enterprise database to the warehouse database
- a list of the columns used as sources for the mapping
- the transformation expression used for each mapping
- the time at which source column data was last updated

Before you can use ETL functionality,

- Informatica’s MX2 API (version 1.2) must be installed on the Intelligence Server host machine. Please consult your System Administrator for details.
- For the specific project, access the Project Configuration Editor: Project definition category and in the ETL Configuration subcategory, specify the ETL settings.
- Grant users the "View ETL Information" privilege.
Introduction

This appendix is a specification of the Desktop Commands used in MicroStrategy products. Even though Desktop Commands can be invoked from the command line, this document focuses on the Desktop Homepage usage and describes the commands from an HTML perspective.

The following topics are discussed:

- Background
- Why would you use Desktop Commands?
- Setting the Desktop homepage
- Viewing the Desktop commands
- Commands
Background

Desktop Commands are a collection of methods that MicroStrategy Desktop, as an application, supports. These commands expose functionality such as executing a report, loading an editor, and so on. You can make full usage of this feature in the Desktop homepage.

Why would you use Desktop Commands?

Desktop commands gives you the flexibility to create your own project homepage and customize it according to your needs and understanding.

In HTML, Desktop commands are written using the anchor element. Anchor elements typically contain a reference to a uniform resource locator (URL). When an anchor is clicked, it performs an operation depending on the URL scheme. For example, a file transfer protocol (ftp) anchor starts a file transfer operation; other schemes are http, gopher, and so on. To differentiate a Desktop command from other anchors we use our own syntax. We call an anchor with this special syntax a Desktop anchor. Following is the HTML specification of a Desktop anchor:

```
<A hRef="dss://Command Parameters"></A>
```

The example above describes the fundamental characteristics of a Desktop anchor. However, any other HTML property can be used in the anchor.
Setting the Desktop homepage

MicroStrategy Desktop is the first window displayed when you log on to MicroStrategy in the desktop environment; it serves as the primary point of access to the editors, dialogs, and wizards that enable the use of MicroStrategy desktop interface functions.

MicroStrategy Desktop can display the selected project contents in a designated HTML home page interface which may contain links to the reports, documents, shortcuts, and so on.

The Desktop homepage is displayed only when you select the project within a project source and not for the objects available within the selected project. The objects within the project are always displayed as folders.

When you open MicroStrategy Desktop and log into a project source [Intelligence Server (3-tier) or project source Direct (2-tier) ] and then you select a project within the project source, you will see the HTML homepage which may display links to the reports, folders, documents, description of the project, and so on. To see an HTML page as shown in step 6 of the following procedure, login as User in MicroStrategy Tutorial.

MicroStrategy Desktop offers you to choose between project homepage and Folder List display. You can choose to enable homepage functionality in Desktop Preferences and My Preferences dialog boxes and can specify whether or not to see a designated HTML file when opening a project. If you choose not to see the HTML page, the Folder List appears instead. You can turn this option on by following these steps:

---

To set an HTML homepage

1. Log in to the MicroStrategy project source containing the project for which you want to set an HTML homepage.

2. From the Tools menu select My Preferences option.
3 In the My Preferences dialog box, select the **Enable project home page functionality** check box.

4 Browse for and locate the HTML file in the **HTML file path** box or use the default homepage location displayed in this box.

   If you have created your own customized HTML page to use in the project, locate the file using **Browse** button. Otherwise, the project will use the default homepage designed by MicroStrategy.

5 Click **OK**.

6 Select the project within the project source. A project homepage will be displayed. For example, in the MicroStrategy Tutorial you will see the HTML homepage which looks like the following:

   ![MicroStrategy homepage](image)

If you cannot see the HTML homepage even after you have enabled the homepage option from the My Preferences dialog box, it is because the project homepage option is not enabled in the Desktop Preferences dialog box. To do this, complete the following procedure.
To set Desktop preferences

1. From the Tools menu, select Desktop Preferences. The Desktop Preferences dialog box opens.

2. In the dialog box, select the Enable project home page functionality option.

3. Click OK.

To work with the homepage functionality, the Enable project home page functionality option should always be enabled in both My Preferences and Desktop Preferences dialog boxes.

Viewing the Desktop commands

In the homepage, Desktop commands exist embedded in the document HTML. When you view the source code of the HTML page, you can see the desktop commands.

To see where in the HTML the Desktop commands are embedded, right-click the HTML page and from the shortcut menu, select View Source. The HTML code gets displayed in your default text editor. Scroll through the code and look for anchor elements starting with the following:

```html
<A hRef="dss://Command Parameters"></A>
```

Desktop commands use unique object ID to execute the commands.

Within a project, you can get an object ID by selecting an object, right-clicking it and then, from the shortcut menu, select Properties. The ID is displayed in the dialog box.
Commands

Desktop supports the following commands:

- **ChangeView**: updates the Desktop view style
- **Editor**: loads a Desktop editor
- **Execute**: executes a report or document definition
- **ExecuteDocument**: executes a document definition
- **ExecuteReport**: executes a report definition
- **Open**: opens a connection to a project source or a session to a project
- **Reset**: closes a connection to a project source or a session to a project
- **Shortcut**: finds and selects a node in the folder list pane of the object browser

The description of each of these commands are in the following sections.

**ChangeView**

The ChangeView command shows or hides the shortcut and folder list panes in the object browser.

```
dss://ChangeView  View
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| View       | The view parameter indicates what operation should be executed. The following operations are supported:  
  - ShowShortcutView  
  - ShowFolderView  
  - HideShortcutView  
  - HideFolderView |
Remarks

The ChangeView command can take a list of operations in a single command. For example, to show shortcuts and hide the folder list use the command: ChangeView ShowShortcutView\HideFolderView.

Example

<A hRef="dss://changeview showshortcutview">ShowShortcuts</A>

<A hRef="dss://changeview showfolderview">Show Folders</A>

<A hRef="dss://changeview hideshortcutview">Hide Shortcuts</A>

<A hRef="dss://changeview hidefolderview">Hide Folders</A>

<A hRef="dss://changeview hidefolderview\showshortcutview">Hide Folders and Show Shortcuts</A>

<A hRef="dss://changeview showfolderview\hideshortcutview">Show Folders and Hide Shortcuts</A>
Editor

The Editor command loads a new instance of a Desktop editor.

dss://Editor EditorName

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EditorName</td>
<td>Indicates the name of the editor to load. The command supports the following editors:</td>
</tr>
<tr>
<td></td>
<td>• Search</td>
</tr>
<tr>
<td></td>
<td>• ReportDefinition</td>
</tr>
<tr>
<td></td>
<td>• DocumentDefinition</td>
</tr>
<tr>
<td></td>
<td>• Prompt</td>
</tr>
<tr>
<td></td>
<td>• Filter</td>
</tr>
<tr>
<td></td>
<td>• Template</td>
</tr>
<tr>
<td></td>
<td>• Metric</td>
</tr>
<tr>
<td></td>
<td>• CustomGroup</td>
</tr>
<tr>
<td></td>
<td>• Consolidation</td>
</tr>
<tr>
<td></td>
<td>• Attribute</td>
</tr>
<tr>
<td></td>
<td>• Fact</td>
</tr>
<tr>
<td></td>
<td>• Hierarchy</td>
</tr>
<tr>
<td></td>
<td>• Transformation</td>
</tr>
<tr>
<td></td>
<td>• Partition</td>
</tr>
</tbody>
</table>

Remarks

The Editor command loads a new editor window in the currently selected project.

Example

<A hRef="dss://editor search">Open Search Editor</A>

Execute

The Execute command loads a viewer for a certain object in metadata. The command takes a list of object IDs and Types. The command supports report and document object types.

dss://Execute ObjID1.ObjType1\ObjID2.ObjType2\...\ObjIDn.ObTypen
Remarks

Use the Execute command when you want to execute a report and a document using a single command.

Example

<A hRef="dss://execute B5C67DFC11D60B5610008CB3D1CEE6A4.3\48CAD4644AB189F763E0EAA22BC0E6DC.55">
Execute: {Profit Forecast 2003, Document (Customer Hierarchy)}
</A>

ExecuteDocument

The ExecuteDocument command loads the document editor. The command can execute one or more documents.

The ExecuteDocument command executes the document only if the current project source is in server connection (3-tier).

\[dss://ExecuteDocument DocumentID1\DocumentID2\DocumentIDn\]
Remarks

The DocumentID parameter can be obtained in the Properties Dialog of the Desktop. The command finds the document in the project that is selected when the command is executed.

Example

```xml
<A hRef="dss://executedocument 3D4DA91C4D20DA7532D4AB848C428031"/>
Execute Document: {Document (My Electronics Dashboard)}
</A>

<A hRef="dss://executedocument 3D4DA91C4D20DA7532D4AB848C428031\0BD252404BB97A2167B085848A40A60B"/>
Execute Document: {Document (My Electronics Dashboard), Document (Product Hierarchy)}
</A>
```

ExecuteReport

The ExecuteReport command runs a report and displays it in grid view. The command can execute one or more reports.

```
dss://ExecuteReport
ReportID1\ReportID2\...\ReportIDn
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentID</td>
<td>The ID of the document definition object. The command takes any number of documents to execute.</td>
</tr>
<tr>
<td>ReportID</td>
<td>The ID of the report definition object. The command takes any number of reports to execute.</td>
</tr>
</tbody>
</table>
**Remarks**

The ReportID parameter can be obtained in the Properties Dialog of the Desktop. The command finds the report in the project that is selected when the command is executed.

The reports are executed using the settings (cache, display view, and so on.) currently selected by the user.

**Example**

```html
<A hRef="dss://executereport BF294AA247895DD9354CA9B296D91D33"> Execute Report: {Electronics Revenue vs. Forecast 2003} </A>

<A hRef="dss://executereport BF294AA247895DD9354CA9B296D91D33\2C3DFFB411D6044FC0008C916B98494F"> Execute Report: {Electronics Revenue vs. Forecast 2003, Electronics Revenue By Region} </A>
```

**Open**

The Open command connects to a project source node in the object browser. It can also take a project ID to open the project node.

```html
<dss://Open\ gin\UserP\assword ProjectSourceName\ProjectID\UserLogin\UserPassword>
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectSourceName</td>
<td>The name of the project source node in the object browser control.</td>
</tr>
<tr>
<td>ProjectID</td>
<td>Optional. ID (GUID) of the project object in the configuration metadata.</td>
</tr>
</tbody>
</table>
The Open command searches for the project source node by name. The search is case sensitive. After the project source node is found it gets expanded. If the user is currently not connected to the project source the command opens the login window.

If the ProjectID parameter is sent, the command finds the project node. You can obtain the project ID of a project using the Project Configuration dialog box in Desktop.

The UserLogin and UserPassword parameters set the default values of the login window when the project source node is expanded. The login window is only displayed if the user is not currently connected to the project source node.

### Example

```xml
<A hRef="dss://open MicroStrategy Tutorial">Open Microstrategy Tutorial</A>

<A hRef="dss://open MicroStrategy Tutorial \B19DEDCC11D4E0EFC000EB9495D0F44F">Open Tutorial Project</A>

<A hRef="dss://open MicroStrategy Tutorial \B19DEDCC11D4E0EFC000EB9495D0F44F\Administrator">Open Tutorial Project using Administrator login</A>
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserLogin</td>
<td>Optional. Login string to use as default in the login window.</td>
</tr>
<tr>
<td>UserPassword</td>
<td>Optional. Password string to use as default in the login window.</td>
</tr>
</tbody>
</table>

### Remarks

The Open command searches for the project source node by name. The search is case sensitive. After the project source node is found it gets expanded. If the user is currently not connected to the project source the command opens the login window.

If the ProjectID parameter is sent, the command finds the project node. You can obtain the project ID of a project using the Project Configuration dialog box in Desktop.

The UserLogin and UserPassword parameters set the default values of the login window when the project source node is expanded. The login window is only displayed if the user is not currently connected to the project source node.
## Reset

The Reset command closes a session to a project or a connection to a project source.

```
dss://Reset ProjectSourceName\ProjectID
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectSourceName</td>
<td>The name of the project source node in the folder list.</td>
</tr>
<tr>
<td>ProjectID</td>
<td>Optional. ID (GUID) of the project object.</td>
</tr>
</tbody>
</table>

### Remarks

If the ProjectID parameter is sent, the command closes the session to the project. Otherwise, the command will close the connection to the project source node.

### Example

```
<A hRef="dss://reset MicroStrategy Tutorial">Close connection to MicroStrategy Tutorial</A>

<A hRef="dss://reset MicroStrategy Tutorial \B19DEDCC11D4E0EFC000EB9495D0F44F">Close session to Tutorial Project</A>
```

## Shortcut

The Shortcut command finds a folder node in the object browser. If the folder is found then it is selected and the contents of the folder are displayed.

```
dss://Shortcut FolderID
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FolderID</td>
<td>The ID of the target folder. You can get the ID using the Properties dialog box in the Desktop.</td>
</tr>
</tbody>
</table>
Remarks

The Shortcut command searches for the folder ID in the project that the user is currently browsing. To select a project in the object browser use the Open command.

The folder ID parameter may specify an special folder name instead of a folder ID. The following special folders are supported:

<table>
<thead>
<tr>
<th>Profile_MyAnswers</th>
<th>Public_Reports</th>
<th>Schema_Subtotals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile_MyFavorites</td>
<td>Public_Prompts</td>
<td>Schema_Particle_Mappings</td>
</tr>
<tr>
<td>Profile_MyObjects</td>
<td>Public_Searches</td>
<td>Schema_Tables</td>
</tr>
<tr>
<td>Profile_MyReports</td>
<td>Public_Metrics</td>
<td>Schema_Hierarchies</td>
</tr>
<tr>
<td>Profile_Objects</td>
<td>Public_Filters</td>
<td>Schema_Functions</td>
</tr>
<tr>
<td>Public_Autostyles</td>
<td>Schema_Objects</td>
<td>Inbox</td>
</tr>
<tr>
<td>Public_Consolidations</td>
<td>Schema_Attributes</td>
<td>Data_Explorer</td>
</tr>
<tr>
<td>Public_Custom_Groups</td>
<td>Schema_Facts</td>
<td>Server_Admin</td>
</tr>
<tr>
<td>Public_Objects</td>
<td>Public_Templates</td>
<td>Schema_Transformations</td>
</tr>
</tbody>
</table>

Example

```<A hRef="dss://shortcut A20C898211D60AE310008BB3b1CfEE6A4">Financial Reports</A>

<A hRef="dss://shortcut profile_myreports">My Reports</A>```
Introduction

An autostyle is a collection of all the formatting layers, allowing you to format different grid units on different report sections. Not every grid unit must be configured to create an autostyle, so any grid unit can have formatting properties set to default. Recall that formats are applied in a particular order, as described in *Order of layers* in Chapter 2, *Reports*. When the lowest layer is set to default, the properties are supplied by the `guiprops.pds` file. This file is saved in the Desktop directory.

This appendix provides the default values for all the formatting properties. Each tab of the Format dialog box is listed below, with the default values for each property on that tab.
**Number**

The default values for the Number tab are:

- Category—general
- Decimal places—zero
- Use thousand separator—yes
- Negative numbers—no special consideration, meaning that neither a red font nor parentheses are used
- Currency symbol—value determined by locale settings
- Currency position—value determined by locale settings
- Format—the number’s data type determines how the value is formatted; for example, a date is formatted differently than a time

**Alignment**

The default values for the alignment properties are:

- horizontal alignment—right
- vertical alignment—top
- text wrapping—no

**Font**

The default font values are:

- Name—value determined by localization settings
- Script—western
- Bold—no
- Italic—no
- Size—10
• Strikeout—no
• Underline—no
• Color—black

**Border**

The values for borders are:

• Top border—yes
• Bottom border—no
• Left border—yes
• Right border—no
• Top border color—black
• Bottom border color—black
• Left border color—black
• Right border color—black

**Patterns**

The pattern defaults are:

• Fill color—white
• Pattern color—blue
• Pattern style—blank
Banding

While this is not a tab on the Format dialog box, banding is enabled by default for the following autostyles:

- Accounting
- Finance
- Colorful
- Greybands

Banding is accessed by selecting Options from the Grid menu.

The merge header cells property, which is found on the Grid menu, is set to false on the autostyles listed above. This property allows element names to be repeated for a unit displayed on a row of a report. For example, Country and Region are displayed on the row axis of a report. If merge header cells is set to true, the report displays as:

```
<table>
<thead>
<tr>
<th>Country</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>Northwest</td>
</tr>
<tr>
<td></td>
<td>Northeast</td>
</tr>
<tr>
<td></td>
<td>South</td>
</tr>
<tr>
<td>Germany</td>
<td>Saxony</td>
</tr>
<tr>
<td></td>
<td>Bavaria</td>
</tr>
</tbody>
</table>
```

If merge header cells is set to false, then the report looks like the following:

```
<table>
<thead>
<tr>
<th>Country</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>Northwest</td>
</tr>
<tr>
<td>USA</td>
<td>Northeast</td>
</tr>
<tr>
<td>USA</td>
<td>South</td>
</tr>
<tr>
<td>Germany</td>
<td>Saxony</td>
</tr>
<tr>
<td>Germany</td>
<td>Bavaria</td>
</tr>
</tbody>
</table>
```
aggregate function  A numeric function that acts on a column of data and produces a single result. Examples include SUM, COUNT, MAX, MIN, and AVG.

aggregate table  A fact table that stores data that has been aggregated along one or more dimensions.

See pre-aggregation.

aggregation  The combining of numeric data at a specific attribute level. The most common function is sum, which creates an additive total.

See also pre-aggregation.

allocation  An optional aspect of a fact extension that allows distribution of values according to a user-defined calculation expression.

Compare degradation.

application-level partition  In application-level partitioning, the application rather than the database server manages the partition tables. MicroStrategy supports two methods of application-level partitioning: metadata partition mapping and warehouse partition mapping.

Compare database-level partition.
application object  MicroStrategy object used to provide analysis of and insight into relevant data. Application objects are developed in MicroStrategy Desktop and they are the building blocks for reports and documents. Application objects include these object types: report, document, template, filter, metric, custom group, consolidation, prompt.

atomic  The lowest level of granularity. Cannot be decomposed into smaller parts.

attribute  A data level defined by the system architect and associated with one or more columns in a data warehouse lookup table. Attributes include data classifications like Region, Order, Customer, Age, Item, City, and Year. They provide a means for aggregating and filtering at a given level.

See also:
- attribute element
- attribute form
- child attribute
- constant attribute
- derived attribute
- parent attribute

attribute-based predictive input metric  A predictive input metric used in data mining that allows attributes to be used as inputs to predictive metrics, regardless of the context in which it is used.

attribute element  A value of any of the attribute forms of an attribute. For example, New York and Dallas are elements of the attribute City; January, February, and March are elements of the attribute Month.
attribute form  One of several columns associated with an attribute that are different aspects of the same thing. ID, Name, Last Name, Long Description, and Abbreviation could be forms of the attribute Customer. Every attribute supports its own collection of forms.

attribute relationship  See relationship.

attribute role  A database column that is used to define more than one attribute. For example, Billing City and Shipping City are two attributes that have the same table and columns defined as a lookup table.

banding  A method of organizing values according to a set of descriptive or meaningful data ranges called buckets. For example, customers in the age ranges of 10–20, 21–30, and 31–40, where each set of ages is a band. Banding is also used for display purposes, where every other row is a different color and the two colors alternate.

Compare consolidation.

base table  A fact table that stores data at the lowest level of dimensionality.

cache  A special data store holding recently accessed information for quick future access. This is normally done for frequently requested reports, whose execution is faster because they need not run against the database. Results from the data warehouse are stored separately and can be used by new job requests that require the same data. In the MicroStrategy environment, when a user runs a report for the first time, the job is submitted to the database for processing. However, if the results of that report are cached, the results can be returned immediately without having to wait for the database to process the job the next time the report is run.
child attribute  The lower-level attribute in an attribute relationship.

See also:

• parent attribute
• relationship

compound attribute  An attribute that has more than one key (ID) form.

compound key  In a relational database, a primary key consisting of more than one database column.

compound metric  A metric that cannot have a level placed on the entire metric, although it can be set separately on each of the components.

compression ratio  The average number of child records combined to calculate one parent record. For example, the compression of ratio between monthly data and yearly data is 12:1. This is used to determine where aggregate tables would have the greatest impact. The larger the compression ratio between two attributes, the more you stand to gain by creating an aggregate table that pre-calculates the higher-level data.

conditional predictive input metric  A predictive input metric used in data mining that allows a metric to be filtered, regardless of the context in which it is used.

configuration object  A MicroStrategy object appearing in the system layer and usable across multiple projects. Configuration objects include these object types: users, database instances, database login IDs, schedules.

consolidation  An object that can be placed on a template and is made up of an ordered collection of elements called consolidation elements. Each element is a grouping of attribute elements that accommodates inter-row arithmetic operations.

Compare custom group.
**consolidation element** A line item in a consolidation based on attribute elements. For example, Year=2002 / Year=2003.

**constant attribute** See implicit attribute.

**custom group** An object that can be placed on a template and is made up of an ordered collection of elements called custom group elements. Each element contains its own set of filtering qualifications.

**dashboard** A popular means of displaying and distributing data from business intelligence projects. Dashboards provide key metrics as well as summary information.

**data definition** Report execution steps that establish how the data is accessed and manipulated in the data warehouse.

**data mining** A technique that is generally used to find hidden predictive information from a large amount of data. This process involves using existing information to gain new insights into business activities by applying predictive models, using analysis techniques such as regression, classification, clustering, and association.

**Data Explorer** A portion of the interface used to browse through data contained in the warehouse. Users can navigate through hierarchies of attributes that are defined by the administrator to find the data they need.
**database-level partition**  In database-level partitioning (sometimes called server-level partitioning), the database server rather than MicroStrategy manages the partitioned tables. The original table is not physically broken into smaller tables. Instead, the database server logically partitions the table according to parameters specified by the database administrator. You do not need to take any action in MicroStrategy to support the partitioning. Since only the logical table is displayed to the end user, the partitioning is transparent to MicroStrategy.

Compare **application-level partitioning**.

**data mart**  1) A database, usually smaller than a data warehouse, designed to help managers make strategic decisions about their business by focusing on a specific subject or department.

2) A database instance used to store result sets saved to data mart tables.

**data mart report**  A special kind of report that saves its report data in a database rather than returning those results to the user. Data mart reports either create a new table in the database to store the report data or append the report data into an existing table.

**data mart table**  A relational table that is used to store the report data from a data mart report.

**data warehouse**  1) A database, typically very large, containing the historical data of an enterprise. Used for decision support or business intelligence, it organizes data and allows coordinated updates and loads.

2) A copy of transaction data specifically structured for query, reporting, and analysis.

**degradation**  A type of fact extension in which values at one level of aggregation are reported at a second, lower attribute level.

Compare **allocation**.
**derived attribute**  An attribute calculated from a mathematical operation on columns in a warehouse table. For example, Age can be calculated from the expression [Current Date–Birth Date].

See also:

- attribute
- implicit attribute

**derived metric**  A metric based on data already available on the report. It is calculated on the Intelligence Server, not in the database. Use a derived metric to perform column math, that is, calculations on other metrics, on report data after it has been returned from the database.

**dimensionality**  See level.

**drill**  A method of obtaining supplementary information after a report has been executed. The new data is retrieved by requerying the Intelligent Cube or database at a different attribute or fact level.

See also:

- page-by
- pivot
- sort
- subtotal
- surf

**dynamic aggregation**  Rollup of metric values that occurs when an attribute is moved from the report grid to the Report Objects. Whenever the attributes in the Report Objects are not the same as the attributes on the grid, dynamic aggregation has occurred. Dynamic aggregation happens on-the-fly, in memory.
**dynamic relationship**  When the relationship between elements of parent and child attributes changes. These changes often occur because of organizational restructuring; geographical realignment; or the addition, reclassification, or discontinuation of items or services. For example, a store may decide to reclassify the department to which items belong.

**entry level**  The lowest level set of attributes at which a fact is available for analysis.

**extraction, transformation, and loading (ETL)**  
1) The process used to populate a data warehouse from disparate existing database systems.
2) Third-party software used to facilitate such a process.

**fact**  
1) A measurement value, often numeric and typically aggregatable, stored in a data warehouse.

2) A schema object representing a column in a data warehouse table and containing basic or aggregated numbers—usually prices, or sales in dollars, or inventory quantities in counts.

See also **metric**.

**fact table**  A database table containing numeric data that can be aggregated along one or more dimensions. Fact tables can contain atomic or summarized data.

Compare:

- **aggregate table**
- **base table**

**filter**  A MicroStrategy object that specifies a set of criteria used to limit the data returned in a report. Specifically, it limits the returned values of an attribute in the result set to a specified range. It is normally implemented in the SQL WHERE clause. Examples include “2002”, “All weekdays in May”, “Stores in the Northeast”.
**formatting layer** The part of a report that allows you to control how a report looks. The basic formatting layers are zones, which are the rows and headers of a report, and grid units, which are the attribute values. Other formatting layers, such as thresholds and subtotals, can be thought of as extensions of these two basic types.

**formatting zone** Determines what formatting is applied to any data or object located in the zone. When an object on a report is moved from one formatting zone to another (as a result of pivoting, for example), the formatting of the object changes based on the new zone.

**Freeform SQL** A MicroStrategy reporting feature that allows you to use your own customized SQL statements to retrieve data from any relational databases that are included in a MicroStrategy project.

**grid unit** The individual attributes, metrics, consolidations, and custom groups that can be placed on a report grid.

**hierarchy** A set of attributes defining a meaningful path for element browsing or drilling. The order of the attributes is typically—though not always—defined such that a higher attribute has a one-to-many relationship with its child attributes.

**HTML document** 1) A compound report displaying multiple grids and graphs. 2) The MicroStrategy object that supports such a report.

**implicit attribute** An attribute that does not physically exist in the database because it is created at the application level. Such an attribute has its expression defined as a constant value, though nothing is saved in a column. For example, you may wish to create columns in the database with a value of 1 for every row to get around COUNT limitations. You do not have to actually create the column, though, because in the Attribute Editor, you can just enter a “1” in the expression to create a count. Implicit attributes are useful in analyzing and
retrieving information. When analyzing data, you can use constant attributes to create a COUNT to keep track of the number of rows returned. You can use constant attributes when building metrics, where you can sum the column holding the constant to create a COUNT. Any constant is acceptable.

Compare derived attribute.

**Intelligent Cube** A copy of the report data saved in memory and used for manipulation of the view definition. This division allows multiple reports with different views to share a common data definition.

**joint children** Joint child relationships are another type of many-to-many relationship where one attribute has a many-to-many relationship to two otherwise unrelated attributes. These relationships can be modeled and conceptualized like traditional attributes, but like facts, they exist at the intersection of multiple attribute levels. For example, consider the relationship between three attributes: promotion, item, and quarter. In this case, promotion has a many-to-many relationship to both item and quarter. An example of a promotion might be a “Red Sale” where all red items are on sale. A business might run this promotion around Valentine’s Day (Q1) and again at Christmas time (Q4).

**key form** One of a set of attribute forms required for unique identification of an element in an attribute. Also called the ID or ID form.

See also attribute form.
**level**

1) In a data warehouse, facts are said to be stored at a particular level defined by the attribute IDs present in the fact table. For example, if a fact table has a Date column, an Item_ID column, and a fact column, that fact is stored at the Date/Item level.

2) With regard to metric calculation, the level is the level of calculation for the metric. For example, a metric on a report with Year and Store attributes would be calculated at the Year/Store level.

See also **level of aggregation**.

**level of aggregation**

The point in an attribute hierarchy where aggregation is performed. For example, in the geographical State--City--Store hierarchy there are three possible levels of aggregation.

**logical data model**

A graphical representation of data that is arranged logically for the general user, as opposed to the physical data model or warehouse schema, which arranges data for efficient database use.

**logical table**

Logical tables are MicroStrategy objects that form the foundation of a schema. Logical tables in the MicroStrategy schema consist of attributes and facts. There are three types of logical tables:

- A logical table is created for each physical table that is imported into a project, using the Warehouse Catalog. This type of logical tables maps directly to physical tables in the data warehouse.

- A logical view does not point directly to a physical table and is defined using a SQL query against the data warehouse.

- A table alias points directly to a physical table. A table alias can have a different name from the physical table.

See also:

- **logical view**
- **table alias**
logical view  One of the three types of logical tables in the MicroStrategy environment. The other two types are logical tables and table aliases. A logical view does not point directly to a physical table and is defined using a SQL query against the data warehouse. Using MicroStrategy, you can create logical views, which can be used in the same way as the logical tables. This means that you define attributes, facts, and other schema objects based on the logical views.

See also logical table.

locked hierarchy  A hierarchy that has at least one attribute that may not be browsed by end users. Application Designers typically lock hierarchies if there are so many attribute elements that element browsing is not usable.

many-to-many  An attribute relationship in which multiple elements of a parent attribute can relate to multiple elements of a child attribute, and vice versa.

See also:
  • one-to-one
  • one-to-many
  • many-to-one
  • relationship

many-to-one  An attribute relationship in which (1) multiple elements of a parent attribute relate to only one element of a child attribute, and (2) every element of the child attribute can relate to multiple elements of the parent.

See also:
  • one-to-one
  • one-to-many
  • many-to-many
  • relationship
**metadata**  A repository whose data associates the tables and columns of a data warehouse with user-defined attributes and facts to enable the mapping of the business view, terms, and needs to the underlying database structure. Metadata can reside on the same server as the data warehouse or on a different database server. It can even be held in a different RDBMS.

**metric**  1) A business calculation defined by an expression built with functions, facts, attributes, or other metrics. For example: \( \text{sum(dollar\_sales)} \) or \([\text{Sales}] - [\text{Cost}]\)

2) The MicroStrategy object that contains the metric definition.

See also **fact**.

**metric-based predictive input metric**  A predictive input metric used in data mining that ensures that the dimensionality of a metric is fixed.

**MOLAP**  Multidimensional online analytical processing.

**multidimensional cache**  Copy of the report data saved in memory. This cache is used for manipulation of the view definition. Also called an Intelligent Cube.

**multidimensional expression**  A query language similar to SQL. MDX is defined by Microsoft. An MDX expression returns a multidimensional result set (dataset) that consists of axis data and cell data. MDX is used to query cubes, which are used by SAP BW to store data. When accessing the data from SAP BW to generate reports, the MicroStrategy Intelligence Server generates MDX.

See also **SAP BW**.

**nonaggregatable metric**  A metric that is not additive along all dimensions. For example, “Stock On Hand at End of Week” is not additive across time: the stock on hand at the end of the week is not the sum of the stock on hand at end of each day in the week.
**one-to-many** An attribute relationship in which every element of a parent attribute can relate to multiple elements of a child attribute, while every element of the child attribute relates to only one element of the parent. The one-to-many attribute relationship is the most common in data models.

See also:
- one-to-one
- many-to-many
- many-to-one
- relationship

**one-to-one** An attribute relationship in which every element of the parent attribute relates to exactly one element of the child attribute, and vice versa.

See also:
- one-to-many
- many-to-one
- many-to-many
- relationship

**page-by** Segmenting data in a grid report by placing available attributes, consolidations, and metrics on a third axis called the Page axis. Since a grid is two-dimensional, only a slice of the cube can be seen at any one time. The slice is characterized by the choice of elements on the Page axis. By varying the selection of elements, the user can page through the cube.

See also:
- drill
- pivot
- sort
• subtotal
• surf

**parent attribute**  The higher-level attribute in an attribute relationship with one or more children.

See also:
• **child attribute**
• **relationship**

**partial relationship**  An attribute relationship in which elements of one attribute relate to elements of a second attribute, while the opposite is not necessarily true.

See also:
• **relationship**
• **one-to-many**
• **many-to-one**
• **many-to-many**

**partition**  A relational database table broken down into smaller component tables. This can be done at the database level or at the application level.

See also:
• **application-level partition**
• **database-level partition**

**partition base table**  A warehouse table that contains one part of a larger set of data. Partition tables are usually divided along logical lines, such as time or geography. Also referred to as a PBT.

See also **partition mapping**.
partition mapping  The division of large logical tables into smaller physical tables based on a definable data level, such as month or department. Partitions minimize the number of tables and records within a table that must be read to satisfy queries issued against the warehouse. By distributing usage across multiple tables, partitions improve the speed and efficiency of database queries.

partition mapping table  A warehouse table that contains information used to identify the partitioned base tables as part of a logical whole. Also referred to as a PMT.

See also:
• partition base table
• partition mapping

physical warehouse schema  A detailed graphic representation of your business data as it is stored in the data warehouse. It organizes the logical data model in a method that makes sense from a database perspective.

See also schema.

pivot  To reconfigure data on a grid report by placing report objects (attributes, metrics, consolidations) on different axes. Also, to reconfigure a grid report by interchanging row and column headers, and hence the associated data. Subset of cross-tab.

See also:
• drill
• page-by
• sort
• subtotal
• surf
PMML An XML standard used to represent data mining models that thoroughly describes how to apply a predictive model. It was developed by the Data Mining Group, DMG, an independent consortium consisting of over two dozen companies including MicroStrategy.

predictive input metric A metric used in data mining that encapsulates the definition of another attribute or metric. There are three types of predictive input metrics: attribute-based input metrics, metric-based input metrics, and conditional input metrics.

pre-aggregation Aggregation, or the calculation of numeric data at a specific attribute level, that is completed before reports are run, with the results stored in an aggregate table.

See also:
- aggregate table
- aggregation

prompt 1) MicroStrategy object in the report definition that is incomplete by design. The user is asked during the resolution phase of report execution to provide an answer that completes the information. A typical example with a filter is choosing a specific attribute on which to qualify.

2) In general, a window requesting user input, as in “type login ID and password at the prompt.”

quality relationship The relationship between a parent attribute and two or more “joint child” attributes. The parent attribute is referred to as a “quality” because its definition is complete only with the intersection of its joint children.

regression A data mining technique that analyzes the relationship between several predictive inputs, or independent variables, and a dependent variable that is to be predicted.
**relationship**  An association specifying the nature of the connection between one attribute (the parent) and one or more other attributes (the children). For example, City is a child attribute of State.

See also:

- parent attribute
- child attribute
- partial relationship
- quality relationship
- one-to-one
- one-to-many
- many-to-one
- many-to-many

**report**  The central focus of any decision support investigation, a report allows users to query for data, analyze that data, and then present it in a visually pleasing manner.

See also:

- filter
- template

**report creation**  The process of building reports from existing, predesigned reports in MicroStrategy Desktop or in MicroStrategy Web.

**report design**  The process of building reports from basic report components using the Report Editor in MicroStrategy Desktop or MicroStrategy Web.
SAP BW  SAP Business Information Warehouse is the business intelligence tool developed by SAP. MicroStrategy’s integration with SAP BW allows users to conduct reporting and analysis with the data from SAP BW. For information on SAP BW, please refer to documentation by SAP; for information on MicroStrategy’s integration with SAP BW, please refer to Chapter 4, Creating OLAP Cube Reports, in this guide.

See also multidimensional expression.

Schema  1) The set of tables in a data warehouse associated with a logical data model. The attribute and fact columns in those tables are considered part of the schema itself.

2) The layout or structure of a database system. In relational databases, the schema defines the tables, the fields in each table, and the relationships between fields and tables.

Schema Object  MicroStrategy object created, usually by a project designer, that relates the information in the logical data model and physical warehouse schema to the MicroStrategy environment. These objects are developed in MicroStrategy Architect, which can be accessed from MicroStrategy Desktop. Schema objects directly reflect the warehouse structure and include attributes, facts, functions, hierarchies, operators, partition mappings, tables, and transformations.

Scorecard  A popular means of displaying and distributing data from business intelligence projects. Scorecards typically follow a specific methodology and are focused on key metrics within a business area.

Shadow Metric  A metric that represents the attribute to be included in a predictive model. It allows an attribute to be used as a predictor.
**shortcut metric**  A metric based on metrics already included in a report. They provide a quick way to add new metrics to that report. Shortcut metrics belong to one of these types: percent-to-total metrics, transformation metrics, rank metrics, and running sum metrics.

**simple metric**  A type of metric that can stand alone or be used as a building block for compound metrics. Simple metrics always contain at least one aggregate function, such as sum or average, applied to a fact, attribute, or another metric. The entire metric can only contain one level.

**smart metric**  A property of a compound metric that allows you to change the default evaluation order. Smart metrics calculate subtotals on the individual elements of the compound metric. For example, a smart metric uses the formula \( \text{Sum(Metric1)}/\text{Sum(Metric2)} \) rather than \( \text{Sum(Metric1/Metric2)} \) when calculating subtotals on a report.

**sort**  Arranging data according to some characteristic of the data itself (alphabetical descending, numeric ascending, and so forth).

See also:
- drill
- page-by
- pivot
- subtotal
- surf

**source system**  Any system or file that captures or holds data of interest.
**subtotal**  A totaling operation performed for a portion of a result set.

See also:
- drill
- page-by
- pivot
- sort
- surf

**surf**  To add filters, attributes, attribute elements, metrics, and functions to existing analysis objects.

See also:
- drill
- page-by
- pivot
- sort
- subtotal

**system hierarchy**  The superset hierarchy containing all attributes in a project. Unlike a browse hierarchy, it is not explicitly created but is automatically deduced by the MicroStrategy platform from all information available to it.

Compare **user hierarchy**.

**table size**  The estimated size of a database table in terms of number of rows.

**table alias**  One type of logical table. A table alias is created outside of the Warehouse Catalog and points directly to a physical table. A table alias can have a different name from the physical table. One physical table can have more than one table alias.

See also **logical table**.
**template**  The data definition portion of the template consists of the group of objects (attribute, metrics, custom groups, and so on) that defines the columns of data to be included in the result set. The layout and format of these objects are defined within the template's view definition.

**threshold**  Used to create conditional formatting for metric values. For example, if revenue is greater than $200, format that cell to have a blue background with bold type.

**transformation**  A schema object that encapsulates a business rule used to compare results of different time periods. Transformations are used in the definition of a metric to alter the behavior of that metric.

**transformation metric**  An otherwise simple metric that takes the properties of the transformation applied to it. For example, a metric calculates total sales. Add a transformation for last year and the metric now calculates last year’s total sales.

**user hierarchy**  Named sets of attributes and their relationships, arranged in specific sequences for a logical business organization. They are user-defined and do not need to follow the logical model. Compare **system hierarchy**.

**view definition**  Report execution steps which represent how the data is viewed and manipulated in the Intelligence Server. The view definition determines how the final report data set generated in the data definition steps is manipulated.
INDEX

A
across a level subtotals 54
add-in function. See custom function.
add-in. See custom function.
advanced qualification
    custom expression 229
    joint element list 231
    relationship filter 229
advanced subtotal
    across a level 54
    by position 55
    group by 55
Aerial perspective 458
aggregate function defined on 527
aggregate table defined on 523
    advantages 520
    base table 523
    compression ratio 527
    effectiveness 527
    integrate into project 527
    logical table size 528
    parent-child relationship 526
    query frequency 525
aggregate-aware 528
aggregation defined on 521
degree of 524
dense 524
dynamic 41, 281, 521
metric 279
sparse 524
Alerter. See threshold.
alias
    attribute column 431
    fact column 406, 410
    metric column 290
    report 17
    table 587, 589
alignment formatting defaults 636
all metric format 95
allocation expression 417
analytical processing 3
application object defined on 5
application, data mart 504
application-level partition defined on 530
apply function 230
ApplyAgg 570
ApplyComparison 571
ApplyLogic 572
ApplyOLAP 571
ApplyRelative 571
ApplySimple 569
association
  drill map 474
  level 475
  remove 477
atomic defined on 523
attribute defined on 422
  Attribute Editor 426
  browse form 434
  child 8
  column alias 431
  component. See report display form and browse form.
compound 435
compound key 435
creating in Project Creation Assistant 614
derived attribute 429
derived expression 429
display 434
element defined on 423
expression 422
form defined on 425
form expression 427
form group 432
heterogeneous mapping 429
implicit, attribute constant 428
joint child relationship 583
many-to-many relationship 575
multiple counting in relationship 577
parent 8
properties 422, 423
qualification 213
relationship defined on 8, 422, 433, 574
report display form 434
role 586
simple expression 428
SQL 430
  system hierarchy 433
  virtual (consolidation) 378
attribute component. See report display form and browse form.
Attribute Creation Wizard 614
Attribute Editor 426, 456
attribute filter, hierarchy 462
attribute form display 434
attribute qualification defined on 533
  attribute-to-attribute qualification 218
merge 223
attribute role defined on 586
  automatic recognition 587, 588
  explicit table alias 587, 589
attribute-based predictive input metric 328, defined on 328
creating 329
automatic attribute role recognition 587
autostyle defined on 97
default formats 635
deploy 97
  find and replace 128
average
  moving 304
  running 304
axis format 94

B
banding defined on 362
count 363
format 95
formatting defaults 638
HTML document 443
points 363
qualification 362
size 362
<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>244</td>
</tr>
<tr>
<td>table</td>
<td>523</td>
</tr>
<tr>
<td>URL</td>
<td>444</td>
</tr>
<tr>
<td>border</td>
<td>637</td>
</tr>
<tr>
<td>break by</td>
<td>222</td>
</tr>
<tr>
<td>browse</td>
<td>464</td>
</tr>
<tr>
<td>form</td>
<td>434</td>
</tr>
<tr>
<td>user hierarchy</td>
<td>464</td>
</tr>
<tr>
<td>by position subtotals</td>
<td>55</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>cache</td>
<td>106</td>
</tr>
<tr>
<td>Intelligent Cube</td>
<td>33</td>
</tr>
<tr>
<td>report</td>
<td>35, 106</td>
</tr>
<tr>
<td>report cache</td>
<td>106</td>
</tr>
<tr>
<td>report view</td>
<td>34, 107</td>
</tr>
<tr>
<td>Catalog options in Project Creation Assistant</td>
<td>611</td>
</tr>
<tr>
<td>category. See hierarchy.</td>
<td></td>
</tr>
<tr>
<td>child attribute</td>
<td>8</td>
</tr>
<tr>
<td>choose from all attributes in a hierarchy prompt</td>
<td>395</td>
</tr>
<tr>
<td>choose from an attribute element list prompt</td>
<td>396</td>
</tr>
<tr>
<td>class. See hierarchy.</td>
<td></td>
</tr>
<tr>
<td>column alias</td>
<td></td>
</tr>
<tr>
<td>attribute</td>
<td>431</td>
</tr>
<tr>
<td>fact</td>
<td>406, 410</td>
</tr>
<tr>
<td>metric</td>
<td>290</td>
</tr>
<tr>
<td>column subtotal format</td>
<td>95</td>
</tr>
<tr>
<td>Command Manager</td>
<td></td>
</tr>
<tr>
<td>creating metric</td>
<td>305</td>
</tr>
<tr>
<td>comparison operator</td>
<td>595</td>
</tr>
<tr>
<td>compound attribute</td>
<td>435</td>
</tr>
<tr>
<td>in Project Creation Assistant</td>
<td>615</td>
</tr>
<tr>
<td>compound key</td>
<td>defined on 435</td>
</tr>
<tr>
<td>compound metric</td>
<td>defined on 236</td>
</tr>
<tr>
<td>count</td>
<td>304</td>
</tr>
<tr>
<td>definition</td>
<td>276</td>
</tr>
<tr>
<td>evaluation order</td>
<td>279</td>
</tr>
<tr>
<td>moving average</td>
<td>304</td>
</tr>
<tr>
<td>moving sum</td>
<td>304</td>
</tr>
<tr>
<td>n-tile</td>
<td>305</td>
</tr>
<tr>
<td>rank</td>
<td>303</td>
</tr>
<tr>
<td>running average</td>
<td>304</td>
</tr>
<tr>
<td>running sum</td>
<td>304</td>
</tr>
<tr>
<td>smart metric</td>
<td>277</td>
</tr>
<tr>
<td>compression ratio</td>
<td>defined on 527</td>
</tr>
<tr>
<td>condition</td>
<td>237</td>
</tr>
<tr>
<td>condition. See filter.</td>
<td></td>
</tr>
<tr>
<td>conditional predictive input metric</td>
<td>defined on 332</td>
</tr>
<tr>
<td>conditional predictive input metrics</td>
<td>331</td>
</tr>
<tr>
<td>conditional predictive inputs metric</td>
<td></td>
</tr>
<tr>
<td>creating</td>
<td>333</td>
</tr>
<tr>
<td>conditionality</td>
<td></td>
</tr>
<tr>
<td>embedding options</td>
<td>273</td>
</tr>
<tr>
<td>metric</td>
<td>272</td>
</tr>
<tr>
<td>conditionality (metric)</td>
<td></td>
</tr>
<tr>
<td>advanced options</td>
<td>273</td>
</tr>
<tr>
<td>configuration object</td>
<td>defined on 6</td>
</tr>
<tr>
<td>Configuration Wizard</td>
<td>609</td>
</tr>
<tr>
<td>consolidation</td>
<td>defined on 377</td>
</tr>
<tr>
<td>custom group comparison</td>
<td>387</td>
</tr>
<tr>
<td>element</td>
<td>380</td>
</tr>
<tr>
<td>element formatting</td>
<td>379</td>
</tr>
<tr>
<td>element from different levels</td>
<td>382</td>
</tr>
<tr>
<td>element from unrelated attributes</td>
<td>383</td>
</tr>
<tr>
<td>elements of the same attribute</td>
<td>381</td>
</tr>
<tr>
<td>evaluation order</td>
<td>124, 384</td>
</tr>
<tr>
<td>evaluation order error example</td>
<td>124</td>
</tr>
<tr>
<td>existing element</td>
<td>383</td>
</tr>
</tbody>
</table>
formatting element 379
import element 383
row level math 379
SQL query 385
virtual attribute 378
consolidation element defined on 380
  existing 383
  from different levels 382
  from the same attribute 381
  from unrelated attributes 383
import 383
constant attribute 428
count 304
creating
  attribute in Project Creation Assistant 614
dashboards 441
  fact in Project Creation Assistant 613
  project 610
  project with Project Creation Assistant 607
cross product join 417
cross-dimensional attribute. See joint child relationship.
cube
  importing 189
  mapping 191
custom expression 229
custom function 243
custom group defined on 360
  banding qualification 362
  benefit 361
  changing element header position 374
  changing the totals position 373
  consolidation comparison 387
data mart tables 507
element 367
element header 367
hierarchical display 368
  Keep Group Structure option 372
  sorting 371
  sorting by item metric values 372
  SQL query 375
Custom Group Editor
  create custom group 360
  header display options 367, 368
custom subtotals 60

D

Dashboards defined on 437
dashboards
  best practices 445
  creating 441
  design parameters 445
  example 449
  gauge-based 448
  implementing 448
  XSL samples 452
data access
  security filter 149
data definition defined on 32
evaluation order 120
report cache 35
Data Explorer defined on 457
data mart defined on 503
  application 504
  custom group 507
  report defined on 503
SQL statements 507
table defined on 504, 505
terminology 504, 505
usage 504
VLDB property 506
data mining 313
  example 352
data mining datasets 323
data mining function
  parameters 347
data model 573
data provider. See project source.
data slice 532
data types 559
  Big Decimal 562
  high-precision data types 562
  warehouse catalog 560
data warehouse defined on 3
database instance 611
database scoring 315
dataset report
  creating for data mining 325
datasets
data mining 323
default drill path 474
Default Warehouse Catalog SQL 603
degradation defined on 416
dense aggregation 524
deployment
  Object Templates directory 105
  Public Objects directory 105
  report 98, 105
  Shared Reports 105
derived
  attribute defined on 429
  fact 409
  metric defined on 297
design view 15, 16, 18
Desktop
  Analyst 99, 101
  anchor 622
  command 622
  Designer 100, 101
detail. See fact.
dimension. See hierarchy.
dimensionality defined on 244
dimensionality. See level.
direct access approach
  overview 171
disallowing 418
display format symbol 293
drill defined on 468
  filter 469
  hierarchy 466
  map. See drill map.
  path. See drill path.
drill map 467, 468
  association 474
  association level 475
  default 474
  default drill path 474
  drill path type 470
  filter properties 471
  grid unit level 476
  priority 474
  project 476
  project level 475
  report level 476
  set name 471
  template level 476
  drill map association 474
    level 475
    remove 477
Drill mode. See page-by.
drill path 468
  default 474
  properties 471
  type 470
  dynamic aggregation defined on 41, 45, 47, 281, 521
    function 50
  dynamic relationship defined on 526
E

element
  attribute element 423
  consolidation 380
embedded
  filter 104
  metric 299
  template 104
embedding options 273
empty object template 115
entity. See hierarchy.
entry level defined on 406
entry point 463
ETL Information option 620
ETL process. See extraction, transformation, and loading process.
evaluation order
  consolidation 384
  consolidation example 124
data set 120
data set example 121, 124
default 119
example 118
incompatibility error example 121, 124
  of compound metric 279
  of derived metric 120
report 118
specified 120
view 120
  view example 121, 124
ExecuteReport 630
explicit table alias 587, 589
export 17
expression map 408
extensible markup language. See XML.
extensible stylesheet language. See XSL.
extraction, transformation, and loading
(ETL) process defined on 2, 619

F

fact defined on 405
  allocation expression 417
column alias 410
creating in Project Creation
  Assistant 613
cross product join 417
degradation defined on 416
derived 409
disallowing 418
degradation defined on 416
template 411
Fact Creation Wizard 420
  fact definition 406, 407
  Fact Editor 420
  fact entry level 406
  fact relation 414
  fact table 406
heterogeneous fact column 410
  implicit 409
  level extension 407, 411
  overview 6
table relation 413
Fact Creation Wizard 420, 613
Fact Editor 420
  fact expression 408
fact table defined on 406
  filter defined on 143, defined on 211
  attribute form qualification 215
  attribute qualification 213
  attribute qualification prompt 220
  attribute-to-attribute qualification 218
break by 222
custom expression 229
drilling and 469
dynamic dates 216
embedded 104
filter object prompt 229
imported filter 217
joint element list 231
logical operator 592
merge attribute qualification 223
metric 272
metric and report interaction 273
metric qualification 221
metric qualification prompt 227
operator 592
pattern operator 598
rank and percent operators 597
relationship filter 226
relationship using advanced qualification 229
report 19
report and metric interaction 273
report object prompt 228
set qualification 220
shortcut 108
shortcut to a filter 104
view 35
filter definition prompt 394, 395
filter operator 592
filtered hierarchy 462
find and replace 127
autostyle 128
metric formatting 129
Report Data Options 128
find. See find and replace.
First (function) 305
first function
example 65
flag 583
font formatting defaults 636
form
attribute form 425
group 432
format
all metrics 95
autostyle 97
axis 94
banding 95
column subtotal 95
default values 635
grid unit 87, 89, 94
layer 87
metric 89, 94
metric color 296
metric display 291
metric on a report 95
number display 292
report 78
report border 95
report metric 95
row subtotal 95
threshold 91, 95
zone 87, 88
formatting layer defined on 87
order of layers 92
formatting zone defined on 88
formula
base 244
join type 284
formula. See compound metric.
Freeform SQL reporting 136
Freeform SQL reports in Report Services documents 141
Freeform SQL reports vs. standard reports 140
fully-qualified URL 443
function
add-in. See custom function.
<table>
<thead>
<tr>
<th>Custom</th>
<th>243</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>305</td>
</tr>
<tr>
<td>Last</td>
<td>305</td>
</tr>
<tr>
<td>OLAP</td>
<td>304</td>
</tr>
<tr>
<td>Plug-in</td>
<td>243</td>
</tr>
<tr>
<td>Syntax</td>
<td>567</td>
</tr>
<tr>
<td>User-defined. See custom function.</td>
<td></td>
</tr>
</tbody>
</table>

**G**

| Graph view | 15 |
| Grid unit  | Defined on 89 |
| Format     | 87, 89, 94 |
| Subtotal format | 90 |
| Grid view  | 15 |
| Group by subtotals | 55 |

**H**

| Heterogeneous formatting of elements | 379 |
| Heterogeneous mapping attribute | 429 |
| Heterogeneous partition mapping | 531 |
| Hierarchical sort | 74 |
| Hierarchy | Defined on 455 |
| Attribute Editor | 456 |
| Attribute filter | 462 |
| Browse | 464 |
| Browse attribute | 464 |
| Display | 460 |
| Drill | 466 |
| Entry point | 463 |
| Filtered | 462 |
| Hierarchy Editor | 457, 458, 466 |
| Hierarchy Viewer | 458 |
| Limited | 461 |
| Locked | 460 |
| Organization | 458 |
| Project Creation Assistant | 456 |
| Structure | 459 |
| System hierarchy | 456 |
| User hierarchy | 457 |
| Hierarchy Editor | 457, 458, 466 |
| Hierarchy Viewer | 458 |
| Homogenous partition mapping | 532, 533 |
| HTML document | Defined on 437 |
| Banding | 443 |
| Creating | 441 |
| Example | 454 |
| Images | 443 |
| Layout | 438 |
| Report characteristics | 442 |
| Stylesheet | 440 |
| View | 442 |
| XML | 439 |
| XSL | 440 |
| XSL samples | 452 |
| HTML Document Editor | 441 |

**I**

| Implicit attribute | Defined on 428 |
| Implicit fact | 409 |
| Initialize project | 610 |
| Inner join | 282 |
| Integer constant in metric | 287 |
| Intelligent Cube | Defined on 33, 106 |
| International support | xxviii |

**J**

| Join | Formula | 284 |
| Inner | 282 |
| Metric | 282 |
| Outer | 282 |
| Report | 282 |
type 284, 287
joint child relationship 583
joint children defined on 433
joint element list 231

K
key performance indicator. See fact.

L
Last (function) 305
level defined on 244
   advanced metric options 255
derived metric 298
   extension 411
   filtering 252
   filtering syntax in Command Manager 308
grouping 246
prompt 400
target 245
level filtering
   absolute syntax in Command Manager 308
   ignore syntax in Command Manager 309
   standard syntax in Command Manager 308
   undefined syntax in Command Manager 309
level grouping
   beginning value syntax in Command Manager 310
   ending value syntax in Command Manager 311
   ignore syntax in Command Manager 311
   standard syntax in Command Manager 310
level grouping syntax in Command Manager 309
limited hierarchy 461
linked filter. See shortcut to a filter.
linked template. See shortcut to a template.
locked hierarchy defined on 460
logical data model defined on 4, 573
logical operator
   exclusion 594, 595
   functional description 592
   intersection 594
   union 593
Logical Table Editor 528
logical table size 528

M
many-to-many relationship defined on 9, 575
mapping type 516
MDX
   object remapping 193
member attribute 515
member expression 515
member table 516
merge attribute qualifications 223
merge header cells 638
metadata defined on 5
metadata database 609
metadata partition mapping
   attribute qualification 533
data slice 532
   overview 531
   versus warehouse partition mapping 534
metric defined on 236
   absolute filter syntax in Command Manager 308
advanced options for conditionality 273
advanced options for level 255
aggregation 279
attribute level 245
base formula 244
beginning value grouping syntax in Command Manager 310
column alias 290
Command Manager syntax 305
compound 236
condition 237
condition embedding options 273
conditionality 272
conditionality advanced options 273
definition of compound 276
definition of simple 241
delimiter syntax in Command Manager 306
derived 38
derived and level 298
difference between simple and compound 240
dimensionality. See level.
dynamic aggregation 281
embedded 299
ending value grouping syntax in Command Manager 311
evaluation order of compound 279
filtering level 252
format 89
format at metric level 94
format at report level 95
formatting 291
formatting at all metrics level 95
formula 236, 242
formula join type 284
function plug-in 243
grouping level 246
ignore filter syntax in Command Manager 309
ignore grouping syntax in Command Manager 309, 310
in view filter 42
join 282, 285
join default 282
join type 284, 287
level 298
level (dimensionality) 237, 244
level (dimensionality) syntax in Command Manager 307, 311
level filtering syntax in Command Manager 308
level grouping syntax in Command Manager 309
level unit syntax in Command Manager 307
nonaggregatable defined on 246
operator syntax in Command Manager 306
overview 9
percent-to-total 300
predictive input 327
qualification 46, 221
rank 302
shortcut 73, 296, 299
simple 236
smart 71, 277
sort hierarchically 74
standard filter syntax in Command Manager 308
standard grouping syntax in Command Manager 310
subtotal 279, 280
subtotal dimensionality 289
target level 245
transformation 237, 275, 297, 302, 513
type 236
undefined filter syntax in Command Manager 309
VLDB property 285
metric and view filter 45
metric formatting
   all metrics level 95
   find and replace 129
   metric level 94
   report level 95
metric qualification 46
   and dynamic aggregation 47
break by 222
definition 221
merge attribute qualification 223
output level 221
overview 26
metric set qualification 221
metric-based predictive input metric 330, defined on 330
creating 330
metrics
   predictive 321
Microcube. See caching and Intelligent Cube.
MOLAP defined on 520
moving
   average 304
   sum 304
multidimensional expressions 169
Multiple Exponential Regression 341
Multiple Linear Regression 341
multiple-key sort 74

N
Narrowcast Server, URLs for 443
nonaggregatable metric defined on 246
non-group function
   moving average 304
moving sum 304
n-tile 305
overview 303
rank 303
running average 304
running sum 304
n-tile 305
null check 288
null checking for Analytical Engine 288
number
   display codes 292
   formatting defaults 636

O
object model
   in MicroStrategy 7i 174
   using SAP direct access 174
object prompt 397
object reuse 106
object template 114
   empty 115
   Object Templates directory 105
object. See attribute.
OLAP BAPI 171
OLAP function 304
one-to-many relationship defined on 8
one-to-one relationship defined on 8
operator
   comparison 595
   filter 592
   logical 592
   pattern 598
   rank and percent 597
outer join 282
outline mode 17
output level 221, 226, 362
Index

P

page-by 16
parent attribute defined on 8
parent-child relationship 526
dynamic 526
overview 574
static 526
partition base table defined on 530, 534
partition mapping defined on 529
application-level 530
attribute qualification 533
data slice 532
heterogeneous 531
homogenous 532, 533
metadata 531, 534
partition base table 534
server-level 530
table defined on 533
types 530
warehouse 533, 534
pass-through expression 230, 566
pattern formatting defaults 637
pattern operator 598
PBT. See partition base table.
percent-to-total metric 300
performance indicator. See fact.
physical warehouse schema defined on 4
pivot 16
Plug-In Wizard 243
PMML 319, defined on 319
PMT. See partition mapping table.
pre-aggregation defined on 522
aggregate table 523
base table 523
compression ratio 527
integrate aggregate table 527
logical table size 528
parent-child relationship 526
query frequency 525
predictive input metric 327
predictive metric 321
using in reports 350
predictive metrics aggregating 349
predictive model creating 335, 342
importing 344, 345
Predictive Model Viewer 351
privilege 101
Desktop Analyst 99, 101
Desktop Designer 100, 101
Web Analyst 99, 101
Web Professional 99, 101
Web Reporter 99, 101
probabilities returning 348
project creation 610
Project Creation Assistant 456, 607, 609
project source 609, 610
prompt defined on 143, defined on 391
choose from all attributes in a hierarchy 395
choose from an attribute element list 396
filter definition 394, 395
level 400
object 397
properties 393
qualify on a metric 396
qualify on an attribute 395
report 102
save options 401
search 393
search object 397
System prompt 402
types of 394
user login system prompt 403
value 399
prompt type filter definition 395
Public Objects directory 105

Q
qualification
  attribute 213
  attribute-to-attribute 218
  banding 362
  metric 221
  metric set 221
  set relationship filter 226
qualify
  attribute prompt 395
qualify on a metric prompt 396
quality. See joint child relationship.
query frequency 525

R
rank 303
rank and percent operators 597
rank metric 302
RDBMS platform
  IBM DB2 AS/400 603
  IBM DB2 OS/390 603
  IBM DB2 UDB 603
  Informix 604
  Microsoft SQL Server 605
  NCR Teradata 606
  Oracle 604
  Red Brick 604
  Sybase Adaptive Server 605
  Sybase IQ 12 605
  Tandem NonStop SQL 606
RDBMS server-level partitioning 530
regression 336, defined on 336
relational database management system 566
relationship
dynamic 526
  many-to-many defined on 9, 575
  one-to-many defined on 8
  one-to-one defined on 8
  static 526
relationship filter 226
  advanced qualification 229
relationship filter output level 226
relationship set qualification 226
relative URL 443
replace. See find and replace.
report defined on 13
  advanced sort 74
  aggregation, dynamic 41
  alias 17
  all metrics formatting 95
  autostyle 97
  autostyle defaults 635
  axis format 94
  banding format 95
  cache 106
  column subtotal format 95
  custom subtotal 60
  data definition 32
  default evaluation order 119
  deployment 98, 105
  derived metric 38
  design view 15, 16, 18
  dynamic aggregation 41, 43, 45, 47
  embedded filter 104
  embedded metric 299
  embedded template 104
  ETL information 620
  evaluation order 118
Index

execution 31, 49
export 17
filter 19
filters vs. limits 22
format 78
format layer order 92
formatting default values 635
formatting layers 87
graph view 15
grid unit format 87, 89, 94
grid view 15
hierarchical sort 74
Intelligent Cube 33
interactive editing 16
join 282
layout. See report view and view definition.
metric 42, 45, 46
metric formatting 94, 95
metric qualification 26, 47
overview 46
multiple-key sort 74
object reuse 106
Object Templates directory 105
outline mode 17
overview 10
page-by 16
pivot 16
privilege 101
prompted 102
Public Objects directory 105
reduce SQL passes 30
report as filter 30
report border format 95
report cache 35, 106
report limit 20
report metric format 95
Report Objects 10, 16, 103
report view 34, 107
row subtotal format 95
save prompted report 401
Shared Reports 105
shortcut metric 73
shortcut to a filter 104, 108
shortcut to a template 104, 109
shortcut to report 30
smart subtotal 71
sort 16, 74
sort, advanced 74
sort, hierarchical 74
sort, multiple-key 74
specified evaluation order 120
SQL optimization 30
SQL view 15
stoplight 17
subtotal 54, 60
subtotal format 90
template 109
Template Dependency Validator 112
threshold 17
threshold format 91, 95
total 54
view definition 32
view filter 16, 35, 42, 43, 45, 46, 47
zone format 87, 88
report as filter 30
report border format 95
report creation defined on 15
Report Data Options
drill 469
evaluation order 120, 385
find and replace 128
metric join type 282
report limit 20
report design defined on 15
report display form 434
Report Editor
- design view 15, 16, 18
- evaluation order 385
- graph view 15
- grid view 15
- SQL view 15
reuse objects 106
row subtotal format 95
running average 304
running sum 304

S

SAP
- metadata model 179
SAP BW object
- characteristic 177
- hierarchy 178
- InfoCube 176
- key figure 177
- query cube 176
- relating to MicroStrategy 179
schema object defined on 5
Schema options in Project Creation Assistant 613
scope of analysis. See Intelligent Cube.
scorecards defined on 437
search for dependents 106
search object 397
security
- setting up 148
security filter defined on 149
defined on 149
segmentation 305
server-level partitioning 530
set qualification
- metric qualification 221
- relationship filter 226
set qualification output level 221, 226
setting up security 148
Shared Reports 105
shortcut
to a filter 104, 108
to a report 30
shortcut metric defined on 299
overview 73, 296
percent-to-total 300
rank 302
transformation 302
transformation metric 297
shortcut to a template 104, 109
Template Dependency Validator 112
simple metric defined on 236
base formula 244
condition 237
conditionality 272
definition 241
dimensionality. See level.
formula 242
level 237, 244
transformation 237, 275
smart metric 71, defined on 277
smart subtotal 71
sort 74
- advanced 74
- hierarchical 74
- multiple-key 74
- report 16
sorting
custom group 371
source system defined on 2
sparse aggregation 524
SQL
- for data mart tables 507
- pass, reduce in report 30
- view 15
SQL query syntax 137
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL support</td>
<td>138</td>
</tr>
<tr>
<td>static relationship</td>
<td>defined on 526</td>
</tr>
<tr>
<td>stoplight</td>
<td>17</td>
</tr>
<tr>
<td>stylesheet</td>
<td>440</td>
</tr>
<tr>
<td>subtotal</td>
<td>defined on 280</td>
</tr>
<tr>
<td>advanced</td>
<td>54</td>
</tr>
<tr>
<td>custom</td>
<td>60</td>
</tr>
<tr>
<td>dimensional</td>
<td>64</td>
</tr>
<tr>
<td>dimensionality aware</td>
<td>289</td>
</tr>
<tr>
<td>format</td>
<td>90, 95</td>
</tr>
<tr>
<td>metric</td>
<td>279</td>
</tr>
<tr>
<td>metric dimensionality</td>
<td>289</td>
</tr>
<tr>
<td>nested function</td>
<td>64</td>
</tr>
<tr>
<td>overview</td>
<td>54</td>
</tr>
<tr>
<td>smart</td>
<td>71</td>
</tr>
<tr>
<td>user-defined</td>
<td>63</td>
</tr>
<tr>
<td>using metrics in formula</td>
<td>64</td>
</tr>
<tr>
<td>sum</td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>304</td>
</tr>
<tr>
<td>moving</td>
<td>304</td>
</tr>
<tr>
<td>summary table</td>
<td>defined on 523</td>
</tr>
<tr>
<td>support</td>
<td></td>
</tr>
<tr>
<td>international</td>
<td>xxviii</td>
</tr>
<tr>
<td>system hierarchy</td>
<td>defined on 456</td>
</tr>
<tr>
<td>System prompt</td>
<td>402</td>
</tr>
</tbody>
</table>

**T**

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td></td>
</tr>
<tr>
<td>alias</td>
<td>587</td>
</tr>
<tr>
<td>data mart</td>
<td>505</td>
</tr>
<tr>
<td>fact table</td>
<td>406</td>
</tr>
<tr>
<td>relation</td>
<td>413</td>
</tr>
<tr>
<td>size</td>
<td>defined on 528</td>
</tr>
<tr>
<td>warehouse tables in Project Creation Assistant</td>
<td>610</td>
</tr>
<tr>
<td>table alias</td>
<td>589</td>
</tr>
<tr>
<td>technical support</td>
<td>xxix</td>
</tr>
<tr>
<td>template</td>
<td>109, 115</td>
</tr>
</tbody>
</table>

**U**

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>443</td>
</tr>
<tr>
<td>usage, data mart</td>
<td>504</td>
</tr>
<tr>
<td>user defined object</td>
<td>See fact expression.</td>
</tr>
<tr>
<td>user hierarchy</td>
<td>defined on 457</td>
</tr>
<tr>
<td>browse</td>
<td>464</td>
</tr>
<tr>
<td>browse attribute</td>
<td>464</td>
</tr>
<tr>
<td>display</td>
<td>460</td>
</tr>
<tr>
<td>drill</td>
<td>466</td>
</tr>
<tr>
<td>entry point</td>
<td>463</td>
</tr>
<tr>
<td>filtered</td>
<td>462</td>
</tr>
</tbody>
</table>
limited 461
locked 460
structure 459
user login system prompt 403
user-defined function. See custom function.
user-defined subtotal 63
example 65
Using attribute forms versus characteristic attributes 435

V
value prompt 399
variable. See compound metric.
variance percentage 302
variance, transformation metric 302
view definition defined on 32
evaluation order 120
Intelligent Cube 33
report view 34
view filter 16, 35, 42, 45, 46, 47
View options in Project Creation Assistant 612
virtual attribute (consolidation) 378
VLDB property
data mart 506
hierarchy 285
integer constant in metric 287
metric join type 287
null check 288
null checking for Analytical Engine 288
overview 285
subtotal dimensionality aware 289
zero check 288
W
warehouse partition mapping overview 533
partition base table 534
partition mapping table 533
versus metadata partition mapping 534
warehouse table in Project Creation Assistant 610
Web
Analyst 99, 101
Professional 99, 101
Reporter 99, 101
X
XML 439
XSL 440
Z
zero check 288
zone format 87, 88
zone formatting subtotal 90