



Siebel Business Analytics Server Administration Guide

Version 7.8.2, Rev A
April 2006

Copyright © 2005, 2006, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

PRODUCT MODULES AND OPTIONS. This guide contains descriptions of modules that are optional and for which you may not have purchased a license. Siebel's Sample Database also includes data related to these optional modules. As a result, your software implementation may differ from descriptions in this guide. To find out more about the modules your organization has purchased, see your corporate purchasing agent or your Siebel sales representative.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS. Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Chapter 1: What's New in This Release

Chapter 2: Administration Tool Basics

Administration Tool User Interface Components	21
Online and Offline Repository Modes	29
Checking the Consistency of a Repository or a Business Model	30
Checking In Changes	31
Setting Preferences	32
Using the Options Dialog Box—General Tab	32
Using the Options Dialog Box—Repository Tab	34
Using the Options Dialog Box—Sort Objects Tab	34
Using the Options Dialog Box—Cache Manager Tab	34
Using the Options Dialog Box—More Tab	35
Setting Permissions for Repository Objects	35
Editing, Deleting, and Reordering Objects in the Repository	37
Displaying and Updating Row Counts for Tables and Columns	37
Using the Browse Dialog Box	38

Chapter 3: Planning and Creating a Siebel Business Analytics Repository

Roadmap for Planning and Setting Up an Analytics Repository	41
Process of Analytics Repository Planning and Design	41
About Repository Planning and Design	41
Planning Your Business Model	42
Identifying the Database Content For The Business Model	47
Guidelines For Designing a Repository	51
Creating a New Analytics Repository File	55

Chapter 4: Creating and Administering the Physical Layer in a Repository

Process of Creating the Physical Layer from Relational Data Sources	58
Importing a Physical Schema from Relational Data Sources	58

Process of Creating the Physical Layer from Multidimensional Data Sources	60
Importing a Physical Schema from Multidimensional Data Sources	61
Setting Up Database Objects	62
About Database Types in the Physical Layer	62
Creating a Database Object Manually in the Physical Layer	62
Specifying SQL Features Supported by a Database	64
Setting Up Connection Pools	65
Creating or Changing Connection Pools	68
Setting Up Connection Pool Properties for Multidimensional Data Sources	75
Setting Up Additional Connection Pool Properties for an XML Data Source	77
Setting Up the Persist Connection Pool Property	79
About Physical Tables	81
Creating and Setting Up Physical Tables	83
Creating and Administering General Properties for a Physical Table	84
Viewing Data in Physical Tables or Columns	85
Creating and Administering Columns and Keys in a Physical Table	86
Setting Up Hierarchies in the Physical Layer for a Multidimensional Data Source	88
Setting Physical Table Properties for an XML Data Source	92
Creating Physical Layer Folders	92
Creating Catalog Folders	93
Creating Schema Folders	93
Using a Variable to Specify the Name of a Catalog or Schema	93
Setting Up Display Folders in the Physical Layer	94
About Physical Joins	95
Defining Physical Foreign Keys and Joins	96
Defining Physical Foreign Keys or Complex Joins with the Joins Manager	97
Defining Physical Joins in the Physical Diagram	97
Using Database Hints	99

Chapter 5: Creating and Administering the Business Model and Mapping Layer in a Repository

About Creating the Business Model and Mapping Layer	103
Creating Business Model Objects	104
Duplicate Business Model and Presentation Catalog	104
Creating and Administering Logical Tables	105
Creating Logical Tables	105
Specifying a Primary Key in a Logical Table	106
Reviewing Foreign Keys for a Logical Table	107

Creating and Administering Logical Columns	107
Creating and Moving a Logical Column	108
Setting Default Levels of Aggregation for Measure Columns	109
Associating an Attribute with a Logical Level in Dimension Tables	110
Creating and Administering Logical Table Sources (Mappings)	111
Creating or Removing a Logical Table Source	111
Defining Physical to Logical Table Source Mappings	113
Defining Content of Logical Table Sources	115
About Dimensions and Hierarchical Levels	119
Process of Creating and Administering Dimensions	119
Creating Dimensions	120
Creating Dimension Levels and Keys	120
Creating Dimensions Automatically	125
Setting Up Dimension-Specific Aggregate Rules for Logical Columns	127
Setting Up Display Folders in the Business Model and Mapping Layer	128
Defining Logical Joins	129
Defining Logical Joins with the Joins Manager	130
Defining Logical Joins with the Business Model Diagram	131
Specifying a Driving Table	132
Identifying Physical Tables That Map to Logical Objects	133

Chapter 6: Creating and Maintaining the Presentation Layer in a Repository

Creating the Presentation Layer in the Repository	135
Presentation Layer Objects	136
Working with Presentation Catalogs	137
Working with Presentation Tables	138
Working with Presentation Columns	139
Using the Alias Tab of Presentation Layer Dialog Boxes	141
Generating an XML File from a Presentation Table	141

Chapter 7: Completing Setup and Managing Repository Files

Process of Completing the Setup for a Repository File	143
Saving the Repository and Checking Consistency	143
Add an Entry in the NQConfig.INI File	144
Create the Data Source	145
Start the Analytics Server	145
Test and Refine the Repository	145
Publish to User Community	146

Importing from Another Repository	146
Querying and Managing Repository Metadata	147
Constructing a Filter for Query Results	150
Comparing Repositories	151
Merging Analytics Repositories	153
Exporting Analytics Metadata to IBM DB2 Cube Views	157
Creating an Analytics Multiuser Development Environment	157
Setting Up an Analytics Multiuser Development Environment	158
Making Changes in an Analytics Multiuser Development Environment	160
Setting Up the Repository to Work with Delivers	166
About the SA System Subject Area	167
Setting Up the SA System Subject Area	167

Chapter 8: Setting Up Mobile Analytics

About Mobile Analytics	173
Mobile Analytics Architecture (Server and Laptop)	177
Installing Mobile Analytics	179
Process of Deploying Mobile Analytics Applications	179
About the Mobile Directory Structure	180
Creating and Testing Tables and Indexes in the SQL Anywhere Database	182
Creating and Storing the Mobile Analytics Repository	184
Assign the Mobile Database Type and Connection Pool	184
Testing and Storing the Mobile Repository	186
Creating the Mobile Web Catalog	187
Defining Sourcing Queries for Mobile Analytics	187
Creating the Mobile Application Configuration File	189
Create the Application Configuration File for Mobile Analytics	190
Data Set Definitions	195
Preparing Mobile Analytics Applications for Deployment	201
Testing and Deploying Mobile Analytics	202
Testing Mobile Analytics Applications	203
Testing the Installation Processes	203
Installing an Application Using Disconnected Analytics Application Manager	204
Installing a Mobile Analytics Application Silently	205
Deleting Mobile Analytics Applications	205

Setting Up Mobile Machines to Operate in Silent Mode	205
Synchronizing Mobile Analytics Applications	206
About Preprocessed Synchronization	206
About Incremental Synchronization	208
Synchronizing Mobile Analytics in the Background (Silent Mode)	210

Chapter 9: Administration Tool Utilities and Expression Builder

Utilities and Wizards	213
Time Series Wizard	213
Synchronize Aliases	214
Replace Columns or Tables Wizard	214
Siebel Business Analytics Event Tables	215
Externalize Strings	215
Rename Wizard	216
Update Physical Layer Wizard	216
Generating Documentation of Repository Mappings	217
Removing Unused Physical Objects	218
Extracting Analytics Metadata Using Dependency Tracking	218
Expression Builder	219

Chapter 10: Setting Up Fragmentation Content for Aggregate Navigation

About Aggregate Navigation	227
Specify Fragmentation Content	227

Chapter 11: Administering the Query Environment

Starting the Analytics Server (Siebel Analytics Server Service)	237
Starting the Server from Windows Services	237
Configuring the Server for Automatic Startup in Windows	238
Running the Analytics Server Startup Script in UNIX	238
Changing the User ID in Which the Analytics Server Runs	239
If the Server Fails to Start	240
Shutting Down the Analytics Server	240
Shutting Down the Server in Windows Services	240
Shutting Down the Server from a Command Prompt in Windows	241
Running the Analytics Server Shutdown Script in UNIX	241
Shutting Down the Analytics Server Using the Administration Tool	242
Getting Users to Connect to the Server	242

Administering the Query Log	242
Administering Usage Tracking	246
Setting Up Direct Insertion to Collect Information for Usage Tracking	247
Setting Up a Log File to Collect Information for Usage Tracking	249
Server Session Management	253
Server Configuration and Tuning	255

Chapter 12:Query Caching in the Analytics Server

About the Analytics Server Query Cache	257
Query Cache Architecture	260
Configuring Query Caching	260
Monitoring and Managing the Cache	262
Purging Cache Programmatically	264
Strategies for Using the Cache	265
Cache Event Processing with an Event Polling Table	268
Setting Up Event Polling Tables on the Physical Databases	268
Making the Event Polling Table Active	271
Populating the Analytics Server Event Polling Table	272
Troubleshooting Problems with an Event Polling Table	272
Making Changes to a Repository	273
Using the Cache Manager	274
Displaying Global Cache Information	275
Purging Cache	275
About the Refresh Interval for XML Data Sources	276

Chapter 13:Connectivity and Third-Party Tools

Configuring Analytics ODBC Data Source Names (DSNs)	279
Third-Party Tools and Relational Data Source Adapters	281
Importing Metadata	281
Using IBM DB2 Cube Views with Siebel Business Analytics	282
About Using IBM DB2 Cube Views with Siebel Business Analytics	282
Generating the Import Files	283
Process of Deploying Cube Metadata	292
ODBC Conformance Level	295

Chapter 14: Using Variables in the Analytics Repository

- Using the Variable Manager 297
 - Using Repository Variables 297
 - About Session Variables 299
 - Creating New Variables 301
- Using Initialization Blocks 302
 - Creating and Editing Initialization Blocks 306
 - Tasks Using the Initialization Block Dialog Box—Variable Tab 310
 - Execution Precedence 311

Chapter 15: Clustering Analytics Servers

- About the Cluster Server Feature 313
- Components of the Cluster Server Feature 313
- Implementing the Cluster Server Feature 314
- Chronology of a Cluster Operation 317
- Using the Cluster Manager 318
 - Viewing Cluster Information 319
 - Managing Clustered Servers 324
- Performance Considerations 324

Chapter 16: Security in Siebel Business Analytics

- Analytics Security Manager 327
 - Working with Users 328
 - Working with Groups 329
 - Importing Users and Groups from LDAP 333
- Authentication Options 336
 - Setting Up LDAP Authentication 337
 - Setting Up External Table Authentication 338
 - Setting Up Database Authentication 340
 - About Delivers and Database Authentication 340
 - Maintaining Analytics Server Internal Authentication 341
 - Order of Authentication 342
 - Bypassing Analytics Server Security 342
- Managing Query Execution Privileges 343

Chapter 17: Using XML as a Data Source for Siebel Business Analytics

- Locating the XML URL 347

Using the Analytics Server XML Gateway	348
Analytics Server XML Gateway Example	350
Accessing HTML Tables	356
Using the Data Mining Adapter	357
Using XML ODBC	361
XML ODBC Example	361
XML Examples	362
83.xml	362
8_sch.xml	363
84.xml	364
Island2.htm	365

Chapter 18:SQL Reference

SQL Syntax and Semantics	367
SELECT Query Specification Syntax	367
SELECT Usage Notes	368
SELECT List Syntax	368
Rules for Queries with Aggregate Functions	370
SQL Logical Operators	374
Conditional Expressions	374
SQL Reference	376
Aggregate Functions	377
Running Aggregate Functions	383
String Functions	388
Math Functions	394
Calendar Date/Time Functions	400
Conversion Functions	408
System Functions	410
Expressing Literals	410

Appendix A: Usage Tracking Data Descriptions and Using the Log File Method

Create Table Scripts for Usage Tracking Data	413
Loading Usage Tracking Tables with Log Files	413
Description of the Usage Tracking Data	414

Appendix B: Pharma Mobile Analytics Administration Reference

Sourcing Reports for Pharma Mobile Analytics	417
--	-----

Data Sets for Pharma Mobile Analytics 419

Appendix C: Mobile Analytics Configuration File Reference

Index

1

What's New in This Release

About Siebel Business Analytics Server Licensing

The programs shown in [Table 1 on page 13](#) require installation and use of the Analytics Server. Your license agreement describes the scope of your program license and therefore your permitted use of the Analytics Server. Some of the functions of the Analytics Server described in this document may be outside the scope of, or not applicable to, your specific program license.

Table 1. Programs Requiring Installation and Use of Analytics Server

Oracle Program
Oracle's Siebel Business Analytics Platform Server-CRM Edition
Oracle's Siebel Marketing Server-CRM Edition
Oracle's Siebel Marketing Server-Enterprise Edition
Oracle's Siebel Usage Accelerator Platform Server

What's New in Siebel Business Analytics Server Administration Guide, Version 7.8.2, Rev. A

[Table 2 on page 13](#) lists changes described in this version of the documentation to support Release 7.8.2 of the software. These changes include the name of this guide and many of the products.

Table 2. Changed Features and Information in Siebel Business Analytics Server Administration Guide, Version 7.8.2, Rev. A

Topic	Description
Data Modeling chapter	Rewrote and reorganized content in Data Modeling chapter and then moved all content to "Process of Analytics Repository Planning and Design" on page 41 in Chapter 3, "Planning and Creating a Siebel Business Analytics Repository" .
Chapter 3, "Planning and Creating a Siebel Business Analytics Repository"	This chapter was renamed from Setting Up a Siebel Analytics Repository.
"Setting Up Connection Pools" on page 65	Updated descriptions of preconfigured connection pools.
"Creating or Changing Connection Pools" on page 68	Updated information about initialization blocks.

Table 2. Changed Features and Information in Siebel Business Analytics Server Administration Guide, Version 7.8.2, Rev. A

Topic	Description
"Example of Creating Sources for Each Level of Aggregated Fact Data" on page 112	Moved topic from "Setting Up Fragmentation Content for Aggregate Navigation" and updated content.
"Defining Content of Logical Table Sources" on page 115	Moved topic from "Setting Up Fragmentation Content for Aggregate Navigation" and combined with like material in "Creating and Administering the Business Model and Mapping Layer in a Repository" . Additionally, added instructions for verifying direct joins to a fact table.
"About WHERE Clause Filters" on page 119	Moved topic from "Setting Up Fragmentation Content for Aggregate Navigation" and updated content.
"Tips For Performance Tuning" on page 145	Added topic.
"Merging Analytics Repositories" on page 153	Revised topic to include more examples and expanded definitions of decision types.
"Synchronizing Mobile Analytics Applications" on page 206	Corrected information for first-time synchronization.
"Synchronizing Mobile Analytics in the Background (Silent Mode)" on page 210	Added cautionary note about the syntax of parameters.
"Synchronize Aliases" on page 214	Corrected procedure for synchronizing aliases.
"Administering the Query Log" on page 242	Expanded the section that explains how to use the query log.
"About Session Variables" on page 299	Updated the description of the Group System Session Variables
"Using Initialization Blocks" on page 302	Updated the table that describes the purpose of initialization blocks that are common to all Siebel Business Analytics applications.
Configuring Operating System Authentication	Removed this section from the chapter "Security in Siebel Business Analytics" on page 327 .
"Using XML ODBC" on page 361	Added a cautionary note about importing ODBC DSNs into the repository.
"Usage Tracking Data Descriptions and Using the Log File Method" on page 413	Updated overview information.

Table 2. Changed Features and Information in Siebel Business Analytics Server Administration Guide, Version 7.8.2, Rev. A

Topic	Description
Configuring Marketing Module Metadata	Moved this chapter to <i>Siebel Marketing Installation and Administration Guide</i> . This material will only be available as online help in the Oracle's Siebel Analytics Administration Tool until the next revision of the <i>Siebel Marketing Installation and Administration Guide</i> .
Creating Sources for Each Logical Table	Removed this topic from "Setting Up Fragmentation Content for Aggregate Navigation" on page 227 . It was a duplicate of a task in "Creating and Administering the Business Model and Mapping Layer in a Repository" on page 103 .

Additional Changes

- Changed the name of Oracle's Siebel Analytics products to Siebel Business Analytics products throughout the book except where the name still exists in the software or user interface.
- Made various quality revisions throughout the book.
- Changed references to the Marketing Module Metadata chapter throughout the book so that they refer to the *Siebel Marketing Installation and Administration Guide*.

What's New in Siebel Analytics Server Administration Guide, Version 7.8.2

[Table 3 on page 15](#) lists changes described in this version of the documentation to support Release 7.8.2 of the software.

Table 3. New Product Features in Siebel Analytics Server Administration Guide, Version 7.8.2

Topic	Description
"Checking the Consistency of a Repository or a Business Model" on page 30 and "Saving the Repository and Checking Consistency" on page 143	Added notes about the enhanced consistency check and new error messages.
"Creating a Database Object Manually in the Physical Layer" on page 62	Updated procedure about creating a database object.
"Creating or Changing Connection Pools" on page 68	Corrected description of Maximum Connections field and added a caution note about using OCI for connecting to Oracle. Added values and definitions for the Isolation level field in the General tab of the Connection Pool dialog box.

Table 3. New Product Features in Siebel Analytics Server Administration Guide, Version 7.8.2

Topic	Description
"Viewing Data in Physical Tables or Columns" on page 85	Corrected caution note about setting the user name and password for a connection pool.
"Creating and Administering Columns and Keys in a Physical Table" on page 86	Added information about the External Name field and updated the graphic and procedure.
"Creating and Moving a Logical Column" on page 108	Changed heading and added procedure for moving a logical column.
"Defining Logical Joins" on page 129	Expanded definition of logical joins.
"Merging Analytics Repositories" on page 153	Corrected the procedure that describes how to merge versions of the repository. Also added descriptions and examples of decision types and available suffixes.
"Configuring the Server for Automatic Startup in Windows" on page 238	Changed procedure.
"Query Caching in the Analytics Server" on page 257 and "About the Analytics Server Query Cache" on page 257	Added information in these topics and added cross references to <i>Siebel Business Analytics Web Administration Guide</i> .
"Using IBM DB2 Cube Views with Siebel Business Analytics" on page 282	Added new section.
"Creating and Editing Initialization Blocks" on page 306	Added information about creating a dedicated connection pool for initialization blocks.
"Using the Data Mining Adapter" on page 357	Updated the way in which the Data Mining Adapter operates and corrected the Configuring the Data Mining Adapter procedure.
"Using a DLL File to call the Data Mining Adapter API" on page 358	Updated procedure to include path to the DLL file and deleted the section that contained the content of the DLL file.
Copying Marketing Metadata from One Repository to Another Repository	Deleted this topic. The Export feature is no longer available.

What's New in Siebel Analytics Server Administration Guide, Version 7.8.1

Table 4 on page 17 lists changes described in this version of the documentation to support Release 7.8.1 of the software.

Table 4. New Product Features in Siebel Analytics Server Administration Guide, Version 7.8.1

Topic	Description
"About Siebel Business Analytics Server Licensing" on page 13	Added information about server licensing.
"Icons and Symbols in the Administration Tool" on page 25	Changed existing and added new repository icons.
"Using the Options Dialog Box—Repository Tab" on page 34	Added a new repository option.
"Using the Options Dialog Box—More Tab" on page 35	Added a new Default diagram zoom option.
"Setting Permissions for Repository Objects" on page 35	Added a new topic about sorting by column in the Permissions dialog box.
"Creating a New Analytics Repository File" on page 55	Added new option for checking global consistency while saving a repository.
"Creating a Database Object Manually in the Physical Layer" on page 62	Added a database property that allows everyone to execute direct database requests.
"Setting Up the Persist Connection Pool Property" on page 79	Added this topic.
"Using Stored Procedures with an Oracle Database" on page 86	Added topic about using stored procedures with Oracle databases.
"Adding a Hierarchy to a Physical Cube Table" on page 89	Changed the topic to include the ability to select a default member because every hierarchy must be associated with a member.
Copy Business Model with Presentation Catalog	Removed this topic from "Utilities and Wizards" on page 213 and replaced with "Duplicate Business Model and Presentation Catalog" on page 104 .
"Associating an Attribute with a Logical Level in Dimension Tables" on page 110 , "Defining Physical to Logical Table Source Mappings" on page 113 , and "Update Physical Layer Wizard" on page 216	Added description of sorting columns in dialog boxes.
"Associating an Attribute with a Logical Level in Dimension Tables" on page 110	Added steps to correct the procedure.

Table 4. New Product Features in Siebel Analytics Server Administration Guide, Version 7.8.1

Topic	Description
"Creating and Administering Logical Table Sources (Mappings)" on page 111	Updated and reorganized content of topics, eliminating duplicate content.
"Creating or Removing a Logical Table Source" on page 111	Corrected Step 1 of the instructions about removing tables from a table source.
"Querying and Managing Repository Metadata" on page 147	Added new right-click feature and description of hidden internal objects.
"Comparing Repositories" on page 151	Added and modified definitions of fields and buttons.
"Merging Analytics Repositories" on page 153	Revised steps for merging repositories, added new button descriptions, and added graphic.
"Setting Up an Analytics Multiuser Development Environment" on page 158	Added new menu, new tasks, and corrected sequence of existing tasks.
"Setting Up the Repository to Work with Delivers" on page 166	Added new topic to explain setup for the SA System subject area.
"Removing Unused Physical Objects" on page 218	Added new topic.
"Extracting Analytics Metadata Using Dependency Tracking" on page 218	Added new topic.
"Expression Builder" on page 219	Described new functionality and updated dialog box graphics.
Collecting More Detailed Information About Queries	Removed this obsolete section from Chapter 11, "Administering the Query Environment."
"Monitoring and Managing the Cache" on page 262	Added instructions for setting up caching attributes for a specific physical table.
"Purging Cache Programmatically" on page 264	Changed the definition of the SAPurgeCacheByTable function.
"Creating and Editing Initialization Blocks" on page 306	Added the Enable check box to the table and a description of the new Enable and Disable menu options.
"Creating and Editing Initialization Blocks" on page 306	Steps in procedure reordered and corrected.
"Adding a New User to a Repository" on page 328	Corrected step 6 in the procedure.
"Importing Users and Groups from LDAP" on page 333	Added instructions for creating additional LDAP servers and added a description of the User Name Attribute Type field.

Table 4. New Product Features in Siebel Analytics Server Administration Guide, Version 7.8.1

Topic	Description
"About Delivers and Database Authentication" on page 340	Added new topic. About how <YourProductName> Delivers works with database authentication.
Creating Target Levels and Adding Segmentation Catalogs and Setting Up conforming Dimension Links	Added instructions for duplicating Marketing objects. The Configuring Marketing Module Metadata chapter has been moved to <i>Siebel Marketing Installation and Administration Guide</i> .
"Conversion Functions" on page 408	Changed the description and syntax of the CAST function.

Additional Changes

- Changed the name of Oracle's Siebel Disconnected Analytics to Oracle's Siebel Mobile Analytics throughout this document.
- Changed the name of Oracle's Siebel Pharma Field Analytics to Oracle's Siebel Pharma Mobile Analytics throughout this document.
- Made various quality revisions throughout the document.

2

Administration Tool Basics

This section provides an overview of the user interface components included in the Administration Tool. The Administration Tool is a Windows application that allows the administrator to create and edit repositories.

NOTE: In this guide, tables of values and descriptions contain only values that need a detailed description. Self-explanatory values such as Show Toolbar do not need to be listed.

Administration Tool User Interface Components

This section includes a description of the following interface components:

- [Main Window in the Administration Tool on page 21](#)
- [Menus in the Administration Tool on page 22](#)
- [Toolbar in the Administration Tool on page 24](#)
- [Keyboard Shortcuts in the Administration Tool on page 24](#)
- [Icons and Symbols in the Administration Tool on page 25](#)
- [Online Help in the Administration Tool on page 29](#)

Main Window in the Administration Tool

The main window of the Administration Tool is a graphical representation of the following three parts of a repository:

- **Physical layer.** Represents the physical structure of the data sources to which the Analytics Server submits queries. This layer is displayed in the right pane of the Administration Tool.
- **Business Model and Mapping layer.** Represents the logical structure of the information in the repository. The business models contain logical columns arranged in logical tables, logical joins, and dimensional hierarchy definitions. This layer also contains the mappings from the logical columns to the source data in the Physical layer. It is displayed in the middle pane of the Administration Tool.
- **Presentation layer.** Represents the presentation structure of the repository. This layer allows you to present a view different from the Business Model and Mapping layer to users. It is displayed in the left pane of the Administration Tool.

Figure 1 shows the three layers of a repository, as described in the preceding list.

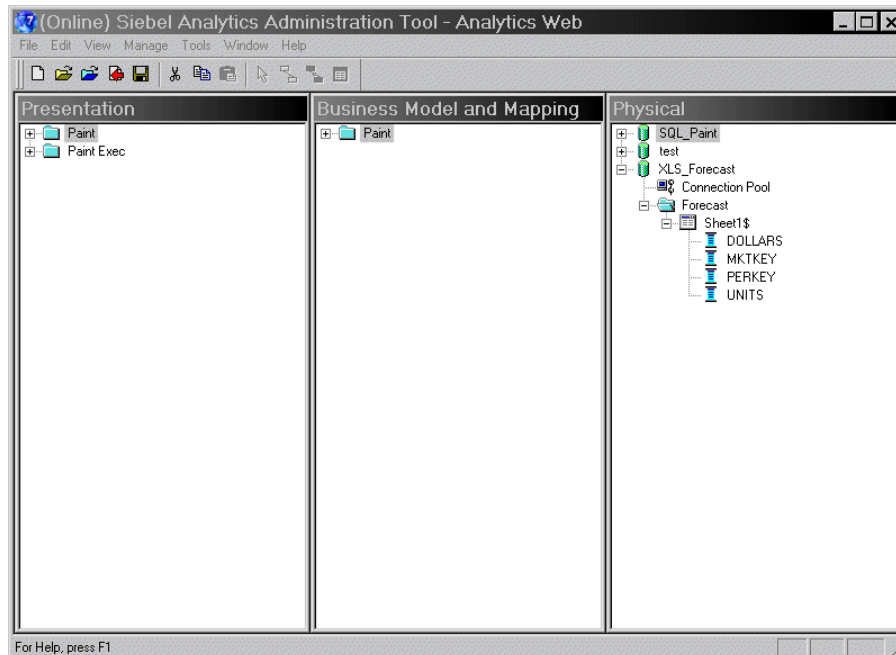


Figure 1. Example Administration Tool Main Window

Menus in the Administration Tool

The Administration Tool includes the following menus:

File

The File menu contains options to work with repositories as well as several server-related options that are active when a repository is open in online mode. Before you open a repository, the File menu has fewer commands available.

Edit

The Edit menu options allow you to edit a repository and work with join conditions, security settings, and repository variables.

View

The View menu options toggle the view of specific metadata panes, give you access to the Join diagrams, and refresh the repository view.

Manage

The Manage menu allows you to access the management functions described in [Table 5](#).

Table 5. Manage Menu Functions

Menu Option	Description
Jobs	This option is available when a repository is opened in online mode. The Job Manager is the management interface to Siebel Scheduler.
Sessions	This option is available when a repository is opened in online mode. In the Session Manager, you can monitor activity on the system, including the current values of repository and session variables.
Cache	This option is available when a repository is opened in online mode and caching is enabled. The Cache Manager allows you to monitor and manage the cache.
Clusters	This option is available when the Analytics Cluster Server feature is installed. The Cluster Manager monitors and manages the operations and activities of the cluster.
Security	The Security Manager displays security information for a repository and provides a central location for security configuration and management.
Joins	The Joins Manager allows you to work with physical and logical joins.
Variables	The Variables Manager allows you to create, edit or delete variables.
Projects	The Project Manager window allows you to create, edit, or remove projects or project elements. Project elements include presentation catalogs, logical fact tables, groups, users, variables, and initialization blocks. You use projects during multiuser development. For more information, see “Setting Up an Analytics Multiuser Development Environment” on page 158.
Marketing	Applies to Oracle’s Siebel Marketing. You need a separate license to use this product.

Tools

The Tools menu options allow you to update all row counts, open the Query Repository dialog box, set options for the Administration Tool, and work with various utilities.

Window

The Window menu options allow you to cascade or tile open layer windows and toggle between them.

Help

The Help menu allows you to obtain the following information:

- Help Topics. Access the online help system for the Administration Tool.
- Siebel on Web. Access the Siebel Web site.
- About Administration Tool. Obtain information about Oracle’s Siebel Administration Tool.

- About Analytics Server. Obtain information about the installed Analytics Server.

Toolbar in the Administration Tool

The toolbar provides access to functions that you use frequently.

To toggle the toolbar on and off

- Select Tools > Options > Show Toolbar.

To dock the toolbar

- Place your cursor on the double bars to the left of the toolbar, and then click and drag to where you want to place the toolbar.

Keyboard Shortcuts in the Administration Tool

Table 6 presents the keyboard shortcuts you can use in the Administration Tool to perform frequent tasks.

Table 6. Keyboard Shortcuts

Menu	Command	Shortcut
File	New	CTRL + N
	Open > Offline	CTRL + F
	Open > Online	CTRL + L
	Save	CTRL + S
	Print	CTRL + P
	Check Global Consistency	CTRL + K
Edit	Cut	CTRL + X
	Copy	CTRL + C
	Paste	CTRL + V
	Delete	DEL
View	Refresh	F5
Tools	Query Repository	CTRL + Q

Icons and Symbols in the Administration Tool

Table 7 presents the icons and symbols used in the Administration Tool with an explanation of what each represents.

Table 7. Icons and Symbols















Icon or Symbol	What It Represents
	A database in the Physical layer and in the Logons tab of the User dialog box. Identifies a database for which you can specify a user ID and a password for the Analytics Server to use.
	A connection pool in the Physical layer.
	A collapsed catalog folder in the Physical layer; a collapsed business model in the Presentation and Business Model and Mapping layers; and a collapsed sources folder within the logical table.
	An expanded catalog folder in the Physical layer; an expanded business model in the Presentation and Business Model and Mapping layers.
	A collapsed schema folder in the Physical layer.
	An expanded schema folder in the Physical layer.
	A physical table in the Physical layer, and a logical table in the Presentation and Business Model and Mapping layers.
	A physical cube table in the Physical layer.
	A table registered as a Siebel event polling table.
	A logical table source in the Business Model and Mapping layer.
	An alias table.
	An object that is a stored procedure call, as specified by the Object Type option in the General tab of the Physical Table dialog box.
	An object that has a Select (view) object type.
	A primary key for a physical or logical table (yellow).

Table 7. Icons and Symbols

Icon or Symbol	What It Represents
	A foreign key for a physical or logical table in the Joins Manager (gray).
	A key for a logical dimension level (blue).
	A key for a multidimensional data source physical level (green).
	A physical or logical complex join in the Joins Manager.
	A dimension in the Business Model and Mapping layer.
	A hierarchy for a multidimensional data source in the Physical layer.
	A level in the Business Model and Mapping layer.
	A level in the Business Model and Mapping layer in which a level key contains one or more columns from another level.
	A level for a multidimensional data source in the Physical layer.
	A physical or logical column.
	A logical column with an aggregation rule.
	A derived logical column.
	A physical cube column from a multidimensional data source. This icon represents columns that are not measures.
	A physical cube column from a multidimensional data source. This icon represents columns that are measures.
	An invalid item. For example, a column may be invalid, if it has no physical mapping.
	A collapsed business model in the Business Model and Mapping layer that is not available for queries.

Table 7. Icons and Symbols
































Icon or Symbol	What It Represents
	An expanded business model in the Business Model and Mapping layer that is not available for queries.
	An item that contains an attribute definition.
	Overlays other symbols to indicate a new item that has not been checked in (appears in online mode only). For example, this icon would appear on top of an alias table icon to indicate an alias table is new.
	A system DSN ODBC entry. Appears in the Import dialog box.
	A measure definition.
	A user.
	A security group.
	An LDAP server.
	A group.
	All users.
	Overlays other icons to indicate an object that is checked out. For example, this icon would appear on top of a table icon to indicate that the table has been checked out.
	A static repository variable.
	A dynamic repository variable.
	A system session variable.
	A nonsystem session variable.
	An initialization block.

Table 7. Icons and Symbols

Icon or Symbol	What It Represents
	A group association (appears only in the results display of the Query Repository dialog box).
	A level-to-level relationship (appears only in the results display of the Query Repository dialog box).
	A type privilege (appears only in the results display of the Query Repository dialog box).
	A query privilege (appears only in the results display of the Query Repository dialog box).
	A privilege package (appears only in the results display of the Query Repository dialog box).
	An object privilege (appears only in the results display of the Query Repository dialog box).
	Overlays other icons to indicate an object that has been cut. Appears with other symbols, for example, to indicate that a cut item is an alias table.
	Target levels (Marketing).
	Qualified list items (Marketing).
	List catalogs (Marketing).
	Conforming dimension links (Marketing).
	Qualifying Key for qualified list items (Marketing).
	Sampling table factor (Marketing).
	Segmentation catalog (Marketing).
	Primary Segmentation catalog (Marketing).

Online Help in the Administration Tool

Most windows and dialog boxes have help buttons that open online help topics containing information to help you complete a task.

Online and Offline Repository Modes

You can open a repository for editing in either online or offline mode. You can perform different tasks based on the mode in which you opened the repository.

This section includes the following topics:

- [Opening a Repository in Offline Mode on page 29](#)
- [Opening a Repository in Online Mode on page 29](#)

Opening a Repository in Offline Mode

Use offline mode to view and modify a repository while it is not loaded into the Analytics Server. If you attempt to open a repository in offline mode while it is loaded into the Analytics Server, the repository opens in read-only mode. Only one Administration Tool session at a time can edit a repository in offline mode.

To open a repository in offline mode

- 1 In the Administration Tool, select File > Open > Offline.
- 2 Navigate to the repository you want to open, and then select Open.
- 3 In the Open Offline dialog box, type a valid user ID and password, and then click OK.

This opens the repository for editing.

NOTE: If the server is running and the repository you are trying to open is loaded, the repository will only open in read-only mode. If you want to edit the repository while it is loaded, you have to open it in online mode. Also, if you open a repository in offline mode and then start the server, the repository will be available to users; any changes you make will become available only when the server is restarted.

Opening a Repository in Online Mode

Use online mode to view and modify a repository while it is loaded into the Analytics Server. The Analytics Server must be running to open a repository in online mode. There are certain things you can do in online mode that you cannot do in offline mode. In online mode, you can perform the following tasks:

- Manage scheduled jobs
- Manage user sessions
- Manage the query cache
- Manage clustered servers

- Stop the Siebel Analytics Server

To open a repository in online mode

- 1 In the Administration Tool, select File > Open > Online.

The Open Online Repository dialog box appears, from which you select a data source.

The data sources displayed in the dialog box are all the User and System DSNs on your computer that are configured using Oracle's Siebel Analytics ODBC driver.

- 2 Select a data source, type a valid user ID and password, and then click OK.

The repository opens that contains the business model corresponding to the data source selected.

NOTE: If you expect to work extensively with the repository (for example, you plan to check out many objects), select the Load all objects option. This loads all objects immediately, rather than as selected. The initial connect time may increase slightly, but opening items in the tree and checking out items will be faster.

Checking the Consistency of a Repository or a Business Model

Repositories in online mode and the business models within them must pass the consistency check before you can make them available for queries. When a repository or business model is inconsistent, a detailed message alerts you to the nature of the inconsistency.

NOTE: The consistency check algorithm was enhanced for Siebel Analytics Version 7.8.2. After upgrading from a previous software version and checking the consistency of your repository, you might see messages that you had not received in previous consistency checks. This typically indicates inconsistencies that had been undetected prior to the upgrade, not new errors.

Consistency check messages are of three types—error, warning, and informational:

- Consistency error messages indicate errors that need to be fixed. Use the information in the message to correct the inconsistency, and then run the consistency check again.
- Consistency warning messages indicate conditions that may or may not be errors, depending upon the intent of the administrator. For example, a warning message about a disabled join may be the result of the administrator intentionally disabling a join, such as eliminating a circular join condition.
- Consistency informational messages provide information about conditions but do not indicate an inconsistency. The following is an example of an informational message:

Fact table does not contain logical key.

To check the consistency of a repository

- 1 In the Administration Tool, select File > Check Global Consistency.

An informational message will alert you if the repository is consistent.

If the repository is not consistent, a more detailed message will display that contains information about the nature of the inconsistency. For example, if you created a business model that does not have a corresponding presentation catalog, a message will alert you and you will be prompted to create one.

- 2 Edit the repository to correct any inconsistencies and run the consistency check again.

To check the consistency of a business model within a repository

- 1 Select a business model and right-click to open the shortcut menu.

- 2 Select the option Check Consistency.

An informational message will alert you if the subject area is consistent.

If the business model is not consistent, a more detailed message appears that contains information about the nature of the inconsistency. For example, if you created a business model that does not have a corresponding presentation catalog, a message will alert you and you will be prompted to create one.

- 3 Edit the business model to correct any inconsistencies and run the consistency check again.

Checking In Changes

When you are working in a repository open in online mode, you will be prompted to perform a consistency check before checking in the changes you make to a repository.

If you have made changes to a repository and then attempt to close the repository without first checking in your changes, the Check In Changes dialog opens automatically. If you move an object from beneath its parent and then attempt to delete the parent, you will be prompted to check in changes before the delete is allowed to proceed.

Use the Check in Changes dialog box to perform the following tasks:

- Make changes available immediately for use by other applications. Applications that query the Analytics Server after you have checked in the changes will see them immediately. Applications that are currently querying the server will see the changes the next time they access any items that have changed.
- Specify that repository changes should be written to disk immediately. If the Analytics Server is shut down abnormally, using the Check Changes dialog box will make sure that checked-in changes to the repository can be recovered. Changes that are checked in but not saved to disk will be restored through the server's error recovery processing. (This processing takes place automatically whenever the Analytics Server has been shut down abnormally.)

To make changes available and have them saved to disk immediately

- In the Administration Tool, select File > Check in Changes.

If the Administration Tool detects an invalid change, an informational message appears to alert you to the nature of the problem. Correct the problem and perform the check-in again.

NOTE: If you make changes to a repository open in online mode, and then attempt to stop the Analytics Server, this option will not be available. This is because your changes will be saved automatically when the server shuts down.

Setting Preferences

You can use the Options dialog box to set preferences for the Administration Tool.

This section includes the following topics:

- [Using the Options Dialog Box—General Tab on page 32](#)
- [Using the Options Dialog Box—Repository Tab on page 34](#)
- [Using the Options Dialog Box—Sort Objects Tab on page 34](#)
- [Using the Options Dialog Box—Cache Manager Tab on page 34](#)
- [Using the Options Dialog Box—More Tab on page 35](#)

Using the Options Dialog Box—General Tab

Use the General tab of the Options dialog box to set general preferences for the Administration Tool.

To set general preferences

- 1 In the Administration Tool, select Tools > Options.
- 2 In the Options dialog box, select the General tab.
- 3 Select the options you want.

The following list contains the options that need additional explanation:

Option	Action When Selected
Tile when resizing	Automatically tiles the layer windows when you resize the Administration Tool.
Display qualified names in diagrams	Makes it easier to identify table sources.
Display original names for alias in diagrams	Makes it easier to identify the actual table referenced.

Option	Action When Selected
Show Time Wizard introduction page	Displays the Time Series Wizard introduction page. The introduction page also contains an option to suppress its display in the future. Use the Time Series Wizard to automate the process of creating measures and calculations to support historical time comparison analyses.
Show Calculation Wizard introduction page	<p>Displays the Calculation Wizard introduction page. The introduction page also contains an option to suppress its display in the future. Use the Calculation Wizard to create new calculation columns that compare two existing columns.</p> <p>You can start the Calculation Wizard by highlighting a logical column in the Business Model and Mapping layer, right-clicking the column to open a shortcut menu, and selecting the option Calculation Wizard.</p>
Check out objects automatically	<p>(online mode only) Automatically checks out the object when you double-click it.</p> <p>NOTE: If the option is not selected, you will be prompted to check out the object before you can edit it.</p>
Show row count in physical view	<p>Displays row counts for physical tables and columns in the Physical Layer. Row counts will not initially display until they are updated. To update the counts, select Tools > Update All Row Counts. You can also right-click on a table or column in the Physical Layer and select the option Update Row Count.</p> <p>Note: Row counts are not shown for items that are stored procedure calls (from the optional Object Type drop-down list in the General tab of the Physical Table dialog). Row counts are not available for XML, XML Server, or multidimensional databases. You cannot update row counts on any new objects until you check them in.</p>
Prompt when moving logical columns	Allows you to ignore, specify an existing, or create a new logical table source for a moved column.
Remove unused physical tables after Merge	Executes a utility to clean the repository of unused physical objects. It might make the resulting repository smaller.
Warn for Merge original repository mismatch	Warns that the original repository is not the origin for the customer and current repositories.

Using the Options Dialog Box—Repository Tab

You can set the following values in the Repository tab:

- **Show tables and dimensions only under display folders.** Administrators can create display folders to organize objects in the Physical and Business Model and Mapping layers. They have no metadata meaning. After you create a display folder, the selected objects appear in the folder as a shortcut and in the database or business model tree as an object. You can hide the objects so that you only see the shortcuts in the display folder.

For more information about creating display folders, see [“Setting Up Display Folders in the Physical Layer” on page 94](#) and [“Setting Up Display Folders in the Business Model and Mapping Layer” on page 128](#).

- **Hide level-based measure.** When selected, level-based measures (columns for which the aggregation rule is other than NONE) will not appear in the Business Model and Mapping layer under the level.

To show tables and dimensions only in display folders

- 1 From the menu bar, choose Tools > Options.
- 2 In the Options dialog box, click the Repository tab.
- 3 Select the Show tables and dimensions only under display folders check box.
- 4 Click OK.

Using the Options Dialog Box—Sort Objects Tab

Use the Sort Objects tab to specify which repository objects appear in the Administration Tool in alphabetical order.

To specify which repository objects appear in alphabetical order

- 1 In the Administration Tool, select Tools > Options.
- 2 In the Options dialog box, select the Sort Objects tab.
- 3 Check the boxes for the objects you want to appear in alphabetical order.

For example, if you want the database objects that appear in the Physical layer to appear in alphabetical order, select the Database check box.

Using the Options Dialog Box—Cache Manager Tab

Use the Cache Manager tab to choose the columns that should be shown in the Cache Manager and the order in which they will be displayed on the Cache tab of the Cache Manager.

To select columns to display in the Cache Manager

- 1 In the Administration Tool, select Tools > Options.
- 2 In the Options dialog box, select the Cache Manager tab.
- 3 Check the boxes for the columns you want display in the Cache Manager.
Clicking on an already checked box removes the check mark. Unchecked items will not appear in the display.
- 4 To change the order of columns in the Cache Manager, select an item, and then use the Up and Down buttons to change its position.

Using the Options Dialog Box—More Tab

Use the More tab to set the speed when scrolling through the trees in various Administration Tool dialog boxes, to set the default window size for the join diagrams, and to specify the path to the multiuser development directory.

To set the scrolling speed and default window size

- 1 In the Administration Tool, select Tools > Options.
- 2 In the Options dialog box, select the More tab.
- 3 Position the cursor on the Scrolling Speed slider to set the speed.
- 4 In the Default diagram zoom drop-down list, you can choose a percentage or Best Fit.
The default window size is Best Fit. If you use the Best Fit option, the following rules apply:
 - If there are five or fewer objects, the zoom level will be 50%.
 - If there are more than five objects, the zoom level changes automatically to Zoom to Fit.

For more information, see [“Setting Up a Pointer to the Multiuser Development Default Directory” on page 160](#).

Setting Permissions for Repository Objects

You can assign user and group permissions to connection pools in the Physical layer and to Presentation layer objects. Additionally, you use Security Manager to set up privileges and permissions. For more information about managing security, see [“Analytics Security Manager” on page 327](#).

The Permissions dialog box displays all currently defined users and groups. For each user and group, you can allow or disallow access privileges for an object by clicking in the check box to toggle among the following options:

- A check mark indicates that a permission is granted.

- An X indicates that a permission is denied.
- An empty check box indicates that a permission has not been modified.

You can access the Permissions dialog box from the following dialog boxes:

- Connection Pool—General tab
- Presentation Catalog Folder—General tab
- Presentation Table—General tab
- Presentation Column—General tab

To add or edit permissions

- 1 In the Permissions dialog box, select the appropriate options for each user and group that you want to modify.
- 2 Click OK.

Sorting Columns in the Permissions Dialog box

There are six ways that you can sort the types and User/Group names. In the Permissions dialog box, there are three columns. The first column contains an icon that represents the type of user or group of users. The second column contains the name of the User/Group object, and the third column contains the Read flag. To change the sort, you can click the heading for the type (blank) and second (User/Group) column.

There are three ways to sort by type and two ways to sort the list of user and group names. This results in a total of six possible sort results ($3 \times 2 = 6$). The following list shows the sort results available by clicking the type column:

- Everyone, Groups, Users (ascending by name of type)
- Users, Groups, Everyone (descending by name of type)
- Type column is in no particular order (Type value is ignored as all names in User/Group column are sorted in ascending order by value in User/Group column)

The following list shows the sort results available by clicking the User/Group column:

- Ascending within the type
- Descending within the type

Examples of Sorting Columns in the Permissions Dialog Box

If you want to sort all rows first by type in ascending order and then, within type, sort the User/Group names in ascending order, use the following steps in the sequence shown:

- 1 Click the blank heading above the type column until the Everyone type appears at the top.
The type column is in ascending order.

- 2 If the User/Group name column is in descending order within each type, click the User/Group heading once.

The list is sorted by type in ascending order and then within type, by User/Group names in ascending order.

- 3 To change the sort by type to descending order, leaving the User/Group names in ascending order, click the type (blank) heading once.
- 4 To change the sort to ignore the type column and sort only on the names in the User/Group column in ascending order, click the type heading once more.
- 5 To continue ignoring the type column and change the sort for the names in the User/Group column to be in descending order, click the User/Group heading.

Editing, Deleting, and Reordering Objects in the Repository

This section contains the standard steps for editing, deleting, and reordering objects. These instructions will not be repeated for each object in the chapters discussing the layers of the repository unless the material is needed to perform a task.

- To edit objects, double-click the object and complete the fields in the dialog box that appears. In some dialog boxes, you can click Edit to open the appropriate dialog box.
- To delete objects, select the object and click Delete.
- To reorder some objects, drag and drop an object to a new location. In some dialog boxes, you can use an up or down arrow to move objects to a new location.

NOTE: Reordering is only possible for certain objects and in certain dialog boxes.

Displaying and Updating Row Counts for Tables and Columns

When you request row counts, the Administration Tool retrieves the number of rows from the physical database for all or selected tables and columns (distinct values are retrieved for columns) and stores those values in the repository. The time this process takes depends upon the number of row counts retrieved.

When updating all row counts, the Updating Row Counts window appears while row counts are retrieved and stored. If you click Cancel, the retrieve process stops after the in-process table (and its columns) have been retrieved. Row counts include all tables and columns for which values were retrieved prior to the cancel operation.

Updating all row counts for a large repository will take a very long time to complete. Therefore, you sometimes might want to update only selected table and column counts. Row counts are not available for the following:

- Stored Procedure object types

- XML data sources and XML Server databases
- Multidimensional data sources
- Data sources that do not support the CountDistinct function, such as Microsoft Access, Microsoft Excel
- In Online mode, Update Row Count will not work with connection pools in which :USER and :PASSWORD are set as the user name and password.

In offline mode, the Set values for variables dialog box appears so that you can populate :USER and :PASSWORD.
- In online mode, after importing or manually creating a physical table or column, the Analytics Server does not recognize the new objects until you check them in. Therefore, Update Row Count will not be available in the menu until you check in these objects.

To display row counts in the Physical layer

- 1 Select Tools > Options.
- 2 In the General tab of the Options dialog box, select the option Show row count in physical view, and then click OK.

To update selected row counts

- 1 In the Physical layer, right-click a single table or column.

You can select multiple objects and then right-click.
- 2 In the shortcut menu, select Update Rowcount.

To update all row counts

- 1 Select Tools > Update All Row Counts.

If the repository is open in online mode, the Check Out Objects window may open.
- 2 Click Yes to check out the objects.

Any row counts that have changed since the last update will be refreshed.

Using the Browse Dialog Box

Use the Browse dialog box to navigate to and select an object to bring into the dialog box from which you entered the Browse dialog box. You can then perform appropriate actions on the object, such as applying constraints to it.

The Browse dialog box is accessible from a number of dialog boxes that allow you to make a selection from among existing objects.

- The left pane of the Browse dialog box contains the following parts:

- A tree listing all of the objects in the Presentation Layer, Business Model and Mapping Layer and the Physical Layer of a repository.
- Tabs at the bottom of the left pane allow you to select a layer. You will see only tabs for the layers that contain objects that can be manipulated in the dialog box from which you entered the Browse dialog box.
- The right pane of the Browse dialog box contains the following parts:
 - Query allows you to query objects in the repository by name and type. The Name field accepts an asterisk (*) as the wild card character, so you can query for partial matches.
 - The Show Qualified Names check box allows you to see to which parents an object belongs.
 - View allows you to view properties of a selected object in read-only mode.

To query for an object in the Browse dialog box

- 1 Select the object type from the Type drop-down list.
- 2 Type the name of the object or a part of the name and the wild card character (*) in the Name field. See the following examples:
 - To search for logical tables that have names beginning with the letter Q, select Logical Tables from the Type drop-down list, and then type Q* in the Name field.
 - To search for logical tables that have names ending with the letters dim, type *dim in the name field.
- 3 Click the Query button.

Relevant objects appear in the query results list.

To select an object in the Browse dialog box

- Select the object you want to select, and then click Select.

The Browse dialog box closes, and the object appears in the previous dialog box.

To synchronize an object in the query results list with the tree control list

- 1 Select an object in the Query list.
- 2 Click the Synchronize Contents icon.

3

Planning and Creating a Siebel Business Analytics Repository

This chapter contains the following topics:

- [Roadmap for Planning and Setting Up an Analytics Repository on page 41](#)
- [Process of Analytics Repository Planning and Design on page 41](#)
- [Creating a New Analytics Repository File on page 55](#)

Roadmap for Planning and Setting Up an Analytics Repository

The roadmap topics are discussed in several chapters in this book. The following are the topics that you will use to plan and set up an analytics repository:

- [Process of Analytics Repository Planning and Design on page 41](#)
- [Creating a New Analytics Repository File on page 55](#)
- [Chapter 4, "Creating and Administering the Physical Layer in a Repository"](#)
- [Chapter 5, "Creating and Administering the Business Model and Mapping Layer in a Repository"](#)
- [Chapter 6, "Creating and Maintaining the Presentation Layer in a Repository"](#)
- [Chapter 7, "Completing Setup and Managing Repository Files"](#)

Process of Analytics Repository Planning and Design

This section contains the following topics:

- [About Repository Planning and Design on page 41](#)
- [Planning Your Business Model on page 42](#)
- [Identifying the Database Content For The Business Model on page 47](#)
- [Guidelines For Designing a Repository on page 51](#)

About Repository Planning and Design

This topics is part of the ["Process of Analytics Repository Planning and Design" on page 41](#).

Before you create a repository or modify the repository that was shipped with Oracle Business Intelligence, you need to plan your business model and develop a design for the repository.

For more information about setting up a repository, see [“Planning and Creating a Siebel Business Analytics Repository” on page 41](#).

An analytics repository has the following layers:

- Repository Physical layer. You create this layer (schema) first from the tables in your physical databases.

CAUTION: Before you create the physical layer of your repository, you must thoroughly analyze and understand your business data model so that you can set up the physical layer correctly. For more information, see [“Planning Your Business Model” on page 42](#).

- Repository Business Model and Mapping layer. After you set up the Physical layer, you can create the Business Model and Mapping layer by dragging and dropping the Physical layer onto the Business Model and Mapping layer. This preserves the mapping that you established in the Physical layer.
- Repository Presentation layer. After you set up the Business Model and Mapping layer, you can create the Presentation layer by dragging and dropping the Business Model and Mapping layer onto the Presentation layer. This layer provides the objects that the end user will access through an application such as Oracle's Siebel Marketing.

Customization involves modifying the Physical, Business Model and Mapping, and Presentation layers. This is a sensitive task and should be done very carefully. It is recommended that you use the information in this chapter when designing and customizing your analytics repository.

Planning Your Business Model

This topic is part of the [“Process of Analytics Repository Planning and Design” on page 41](#).

Planning your business model is the first step in developing a usable data model for decision support. After you have followed the planning guidelines in this section, you can begin to create your repository. This section contains the following topics:

- [Analyzing Your Business Model on page 42](#)
- [Identifying the Content of The Business Model on page 43](#)

Analyzing Your Business Model

Your first task is to thoroughly understand your business model. You have to understand what business model you want to build before you can determine what the physical layer needs to have in it.

In a decision support environment, the objective of data modeling is to design a model that presents business information in a manner that parallels business analysts' understanding of the business structure. A successful model allows the query process to become a natural process by allowing analysts to structure queries in the same intuitive fashion as they would ask business questions. This model must be one that business analysts will inherently understand and that will answer meaningful questions correctly.

This requires breaking down your business into several components to answer the following questions:

- What kinds of business questions are analysts trying to answer?
- What are the measures required to understand business performance?
- What are all the dimensions under which the business operates?
- Are there hierarchical elements in each dimension and what are the parent-child relationships that define each hierarchy?

After you have answered these questions, you can identify and define the elements of your business model.

Identifying the Content of The Business Model

To create a business model, the data needs to be mapped logically to a business model. The Analytics Server can use dimensional models for this purpose. This section discusses some of the components and variants of representative dimensional models.

Businesses are analyzed by relevant dimensional criteria, and the business model is developed from these relevant dimensions. These dimensional models form the basis of the valid business models to use with the Analytics Server. All dimensional models build on a star schema. That is, they model some measurable facts that are viewed in terms of various dimensional attributes.

After you analyze your business model, you need to identify the specific logical tables and hierarchies that you need to include in your business model. For more information about these objects, see [Identifying the Content of The Business Model on page 43](#). The following sections are discussed in this section:

- [Identifying Fact Tables on page 43](#)
- [Identifying Dimension Tables on page 44](#)
- [Identifying Bridge Tables on page 45](#)
- [Identifying Dimension Hierarchies on page 45](#)
- [About Star and Snowflake Models on page 46](#)

Identifying Fact Tables

A fact table is a table with measures. Measures must be defined in a logical fact table. Measures, or facts, are typically calculated data such as dollar value or quantity sold, and they can be specified in terms of dimensions. For example, you might want to see the sum of dollars for a given product in a given market over a given time period.

Each measure has its own aggregation rule such as SUM, AVG, MIN, or MAX. A business might want to compare values of a measure and need a calculation to express the comparison. Also, aggregation rules can be specific to particular dimensions. The Analytics Server allows you to define complex, dimension-specific aggregation rules.

The Analytics Server recognizes that when a table in the Business Model and Mapping layer of a repository has only many-to-one (N:1) logical joins pointing to it, it is a fact table. To see an example of this in a business model, open a repository in the Administration Tool, right-click on a fact table, and select Business Model Diagram > Whole Diagram.

NOTE: The fact table is at the end of a 0, 1:N (zero, one-to-many relationship) join.

Figure 2 on page 44 illustrates the many-to-one joins to a fact table in a Business Model Diagram. In the diagram, all joins have the crow's feet symbol (indicating the many side) pointing into the Fact-Pipeline table and no joins pointing out of it.

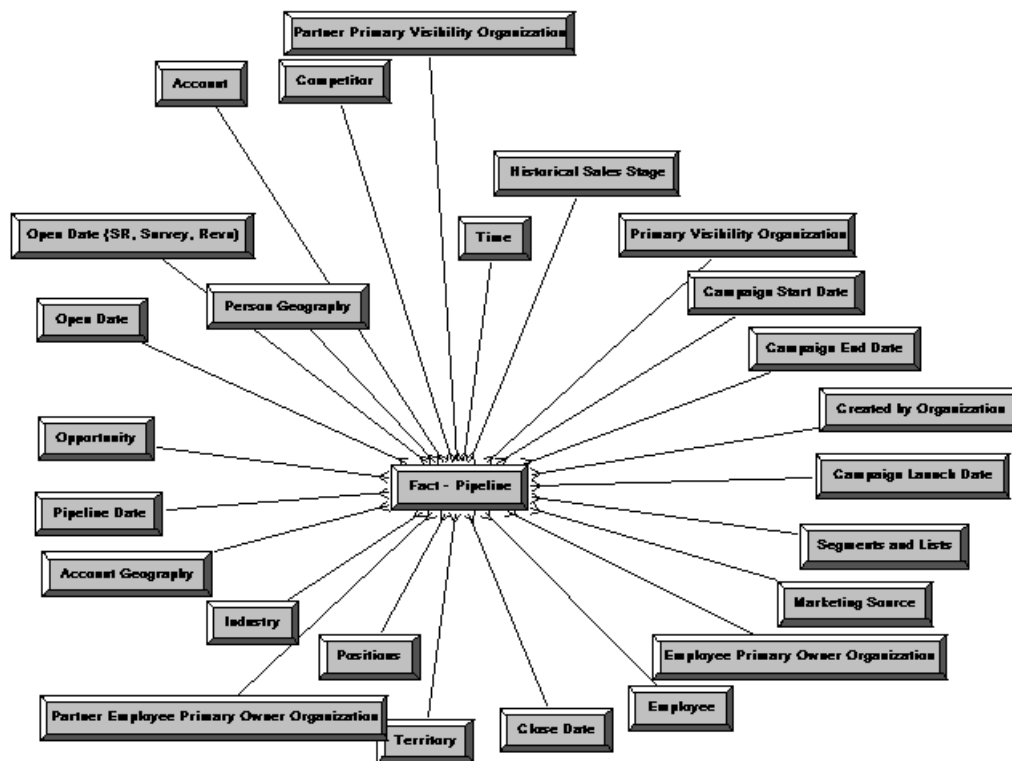


Figure 2. Diagram of Fact Table Joins

NOTE: A bridge table is an exception to this joins rule. For more information, see "Identifying Bridge Tables" on page 45.

Identifying Dimension Tables

A business uses facts to measure performance by well-established *dimensions*, for example, by time, product, and market. Every dimension has a set of descriptive attributes. Dimension tables contain attributes that describe business entities. For example, Customer Name, Region, Address, Country and so on. Dimension tables also contain primary keys that identify each member.

Dimension table attributes provide context to numeric data, such as number of Service Requests. Attributes stored in this example dimension include Service Request Owner, Area, Account, and so on.

The best method to identify dimensions and their attributes is to talk with the analysts in the organization who will use the data. The terminology they use and understand is important to capture.

Identifying Bridge Tables

When you need to model many-to-many relationships between dimension tables and fact tables, you can create a bridge table that resides between the fact and the dimension tables. A bridge table stores multiple records corresponding to that dimension. In the Administration Tool, General tab in the Logical Table dialog box contains a check box that you can select to specify that a table is a bridge table.

Identifying Dimension Hierarchies

Understanding the hierarchies you need in your business is essential to provide the metadata that allows the Analytics Server to determine if a particular request can be answered by an aggregate that is already computed. For example, if month rolls up into year and an aggregate table exists at the month level, that table can be used to answer questions at the year level by adding up all of the month-level data for a year.

A hierarchy is a set of parent-child relationships between certain attributes within a dimension. These hierarchy attributes, called levels, roll up from child to parent; for example, months can roll up into a year. These rollups occur over the hierarchy elements and span natural business relationships.

Dimensions are categories of attributes by which the business is defined. Common dimensions are time periods, products, markets, customers, suppliers, promotion conditions, raw materials, manufacturing plants, transportation methods, media types, and time of day. Within a given dimension, there may be many attributes. For example, the time period dimension can contain the attributes day, week, month, quarter, and year. Exactly what attributes a dimension contains depends on the way the business is analyzed.

A dimensional hierarchy expresses the one-to-many relationships between attributes. Given a sample time dimension, consider the hierarchies it implies, as shown in [Figure 3 on page 45](#).



Figure 3. Sample Hierarchy

With this sample time dimension, days may aggregate, or roll up, into weeks. Months may roll up into quarters, and quarters into years. When one attribute rolls up to another, it implies a one-to-many relationship. The total of all the hierarchy definitions in this sample time dimension make up this time dimension.

These hierarchy definitions have to be specific to the business model. One model may be set up where weeks roll up into a year, and another where they do not. For example, in a model where weeks roll up into a year, it is implied that each week has exactly one year associated with it; this may not hold true for calendar weeks, where the same week could span two years.

Some hierarchies might require multiple elements to roll up, as when the combination of month and year roll up into exactly one quarter. The Analytics Server allows you to define the hierarchy definitions for your particular business, however complex, assuring that analyses will conform to your business definitions.

You should identify as many natural hierarchies as possible. As with business questions, some hierarchies are obvious, but some are not and are only known by the end users who interact with particular aspects of the business. You should verify that you understand the hierarchies so you can define them properly using the Administration Tool.

About Star and Snowflake Models

Star and snowflake models follow the dimensional rules of one-to-many relationships. Star schemas have one-to-many relationships between the logical dimension tables and the logical fact table. Snowflake schemas have those same types of relationships, but also include one-to-many relationships between elements in the dimensions.

NOTE: It is recommended that you minimize the use of snowflake schemas.

For example, in [Figure 4 on page 46](#) and [Figure 5 on page 47](#), Sales Facts and Facts are fact tables and Markets, Periods, Products, Account Hierarchy, Account Region Hierarchy and Account are dimension tables. Any logical table that is not a fact table or a bridge table is a dimension table. For more information about bridge tables, see ["Identifying Bridge Tables" on page 45](#).

[Figure 4 on page 46](#) illustrates a star schema:

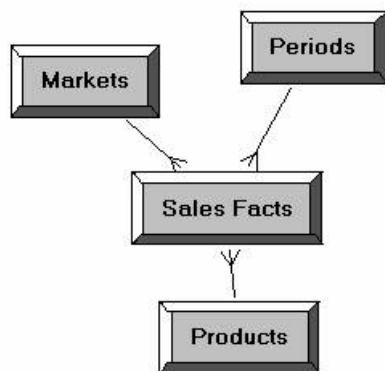


Figure 4. Diagram of a Star Schema

[Figure 5 on page 47](#) illustrates a logical snowflake schema that defines hierarchies by using multiple dimension tables.

NOTE: It is recommended that you minimize the use of snowflake schemas.

The value of the snowflake schema is that the server treats outer joins differently from outer joins that are within a logical table source. Within a logical table source, the joins are always executed. When between logical tables, the joins are only performed when required. For more information about snowflake schemas, see [“About Types of Physical Schemas” on page 47](#).

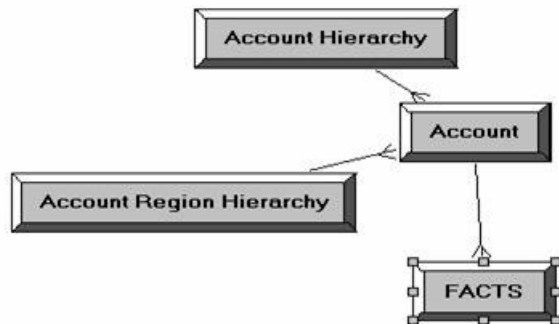


Figure 5. Diagram of Hierarchies Using Multiple Dimension Tables

Identifying the Database Content For The Business Model

This topic is part of the [“Process of Analytics Repository Planning and Design” on page 41](#).

The Analytics Server provides an interface that allows you to map the analytics repository to your underlying physical databases. Sometimes you can control the physical design of the underlying databases. However, sometimes the database already exists and you need to work with what is there. In either case, you need to understand the structure and content of your physical databases.

This section discusses the following topics:

- [About Types of Physical Schemas on page 47](#)
- [About Primary Key-Foreign Key Relationships on page 50](#)
- [Identifying the Database Table Structure To Import on page 50](#)

About Types of Physical Schemas

There are two types of physical schemas (models): entity-relationship (E-R) schemas and dimensional schemas. E-R schemas are designed to minimize data storage redundancy and optimize data updates. Dimensional schemas are designed to enhance understandability and to optimize query performance.

- **Entity-Relationship (E-R) Schemas.** The entity-relationship (E-R) schema is the classic, fully normalized relational schema used in many online transaction processing (OLTP) systems. The relationships between tables signify relationships between data, not necessarily business relationships.

Typically, E-R schemas have many tables, sometimes hundreds or even thousands. There are many tables because the data has been carefully taken apart (normalized, in database terminology) with the primary goal of reducing data redundancy and bringing about fast update performance. E-R schemas are very efficient for OLTP databases. When E-R databases are queried, joins are usually predetermined and can be optimized. E-R databases are usually queried by applications that know exactly where to go and what to ask. These applications typically query small units of information at a time, such as a customer record, an order, or a shipment record.

E-R schemas generally do not work well for queries that perform historical analysis due to two major problems: poor performance and difficulty in posing the question in SQL.

- Performance problems persist with historical E-R queries because the queries require the database to put the data back together again; this is a slow, complex process. Furthermore, because the cost-based optimizers in database management systems are not designed for the level of complexity in such a query, they can generate query plans that result in poor performance.
- A Database Analyst (DBA) who is very familiar with the data might be able to write a SQL query against an E-R schema that can theoretically answer a business question, but such detailed knowledge of the data is generally beyond the scope of the end-user business analyst. Even when the SQL is crafted properly, there is often an unacceptably high response time in returning results to the user.
- **Dimensional Schemas.** A dimensional schema is a denormalized schema that follows the business model. Dimensional schemas contain dimension tables and fact tables. Dimension tables contain attributes of the business, and fact tables contain individual records with a few facts and foreign keys to each of the dimension tables.

Dimensional schemas are used for business analysis and have two major advantages over E-R schemas for decision support:

- Better query performance
- Easier to understand

Dimensional schemas are not nearly as efficient as E-R schemas for updating discrete records, but they work well for queries that analyze the business across multiple dimensions.

The following are two types of dimensional schemas:

- **Star schema.** A star schema is a dimensional schema with a single fact table that has foreign key relationships with several dimension tables.
 - The dimension tables mirror the business model and contain columns with descriptive attributes, such as Product, Size, and Color in the sample Products dimension. Dimension tables also have a key column (or columns) that uniquely identifies each row in the table.

- The fact table has a multipart primary key, often made up of the foreign key references to the dimension tables. The fact table also contains all the measures, or facts, used to measure business performance. The lowest level of detail stored is the granularity of the fact table. Information at higher levels of aggregation is either calculated from the detail level records or precomputed and stored in separate aggregate fact tables, resulting in a multiple-star schema. For a discussion of aggregate fact tables, read [“Identifying Fact Tables” on page 43](#).
- **Snowflake schema.** A snowflake schema is a dimensional schema where one or more of the dimensions are normalized to some extent.

NOTE: It is recommended that you minimize the use of snowflake schemas.

The difference between the type of normalization in a snowflake schema and in an E-R schema is that the snowflake normalization is based on business hierarchy attributes. The tables snowflaked off the dimensions have parent-child relationships with each other that mimic the dimensional hierarchies. In a snowflake schema, multiple logical tables are considered a single logical table.

For example, [Figure 6 on page 49](#) contains a snowflake schema showing that Product Line > Products is a branch of the snowflake.



Figure 6. Diagram of a Snowflake Schema

One dimension hierarchy should be created for the Products branch. The following is a list of the minimal levels that should be created for this hierarchy:

- Grand Total Level
- Detail Level of the dimension that is not joined to the fact table. In this case, it is ProductLine.
- Detail Level of the dimension that is joined to the fact table.

[Figure 7 on page 49](#) shows the hierarchy that should be created.

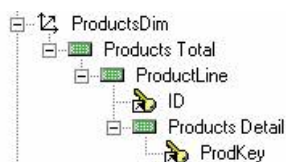


Figure 7. Hierarchy for the Products Branch in the Business Model and Mapping Layer

About Primary Key-Foreign Key Relationships

To fully understand the structure and content of your physical databases, it is important to become familiar with the concepts behind primary key-foreign key relationships.

A primary key-foreign key relationship defines a one-to-many relationship between two tables in a relational database. A foreign key is a column or a set of columns in one table that references the primary key columns in another table. The primary key is defined as a column (or set of columns) where each value is unique and identifies a single row of the table.

Consider [Figure 8](#), where the upper table is a fact table named Sales and the lower table is a dimension table named Date. The Sales fact table contains one row for every sales transaction, and the Date dimension table contains one row for every date the database will potentially cover.

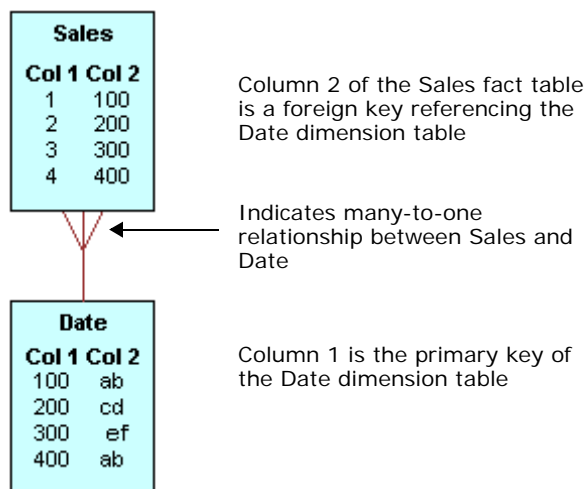


Figure 8. Primary Key-Foreign Key Sample

Because of this primary key-foreign key relationship, you can join the Sales and Date tables to combine the other attributes of the Date table with the records in the Sales table. For example, if an analyst asks for the total sales, the day of the week, the product name, and the store in which the product was sold, the information is compiled by joining the sales, date, product, and store tables through the primary key-foreign key relationships between the Sales table and the various dimension tables.

Identifying the Database Table Structure To Import

The Administration Tool provides an interface to map logical tables to the underlying physical tables in the database. Before you can map the tables, you need to identify the contents of the physical database as it relates to your business model. To do this correctly, you need to identify the following contents of the physical database:

- Identify the contents of each table
- Identify the detail level for each table

- Identify the table definition for each aggregate table. This allows you to set up aggregate navigation. The following detail is required by the Analytics Server:
 - The columns by which the table is grouped (the aggregation level)
 - The type of aggregation (SUM, AVG, MIN, MAX, or COUNT)

For information on how to set up aggregate navigation in a repository, see [“Setting Up Fragmentation Content for Aggregate Navigation” on page 227](#).

- Identify the contents of each column
- Identify How each measure is calculated
- Identify the joins defined in the database

To acquire this information about the data, you could refer to any available documentation that describes the data elements created when the database was implemented, or you might need to spend some time with the DBA for each database to get this information. To fully understand all the data elements, you might also need to ask people in the organization who are users of the data, owners of the data, or the application developers of applications that create the data.

Guidelines For Designing a Repository

This topics is part of the [“Process of Analytics Repository Planning and Design” on page 41](#).

After analyzing your business model needs and identifying the database content your business needs, you can complete your repository design. This section contains some design best practices that will help you implement a more efficient repository.

Typically, you should not make performance tuning changes until you have imported and tested your databases. This would occur during the final steps in completing the setup of your repository. For more information about these final steps, see [“Completing Setup and Managing Repository Files” on page 143](#).

The following topics are discussed in this section:

- [General Tips For Working on the Repository on page 51](#)
- [Design Tips For the Physical Layer \(Schema\) on page 52](#)
- [Design Tips for the Business Model and Mapping Layer on page 53](#)
- [Design Tips For the Presentation Layer on page 55](#)

General Tips For Working on the Repository

The Analytics Server stores metadata in repositories. The administrator uses the graphical user interface (GUI) of the Administration Tool software to create and maintain repositories. An analytics repository consists of three layers. Each layer appears in a separate pane in the Administration Tool user interface and has a tree structure (similar to the Windows Explorer). These layers are not visible to the end user.

Most windows and dialog boxes have online help containing information to help you complete a task. To access a help topic, click the help button in a dialog box or select Help from some menus.

The following list contains some recommended design strategies for the analytics repository structure:

- Perform the majority of repository editing in offline mode to make your results more consistent. In some environments, this might save time during the development effort.

If you work in Online mode, save backups of the online repository before and after every completed unit of work. If needed, use the Copy As feature on the File menu to make an offline copy containing the changes.
- For design independence, import databases for each business model separately. For example, you might have one database for each of the following: Oracle's Siebel Data Warehouse, Oracle's Siebel BBB Data Warehouse, and Oracle's Forecasting Siebel Data Warehouse. Some tables exist in more than one database and you can customize each table for a different business model. This eliminates the need for cross-database joins.
- After importing a database, test it before importing another. Make sure the metadata is generating the correct record set before performing performance tuning activities.
- Use the Physical diagrams in the Administration Tool to verify sources and joins.

Design Tips For the Physical Layer (Schema)

The Physical layer contains information about the physical data sources. The most common way to create the schema in the Physical layer is by importing metadata from databases and other data sources. If you import metadata, many of the properties are configured automatically based on the information gathered during the import process. You can also define other attributes of the physical data source, such as join relationships, that might not exist in the data source metadata.

There can be one or more data sources in the Physical layer, including databases and XML documents. For more information about supported databases, see *System Requirements and Supported Platforms* on Oracle's Siebel SupportWeb.

For each data source, there is at least one corresponding connection pool. The connection pool contains data source name (DSN) information used to connect to a data source, the number of connections allowed, timeout information, and other connectivity-related administrative details. For more information, see ["Setting Up Connection Pools" on page 65](#) and [Chapter 4, "Creating and Administering the Physical Layer in a Repository."](#)

The following is a list of tips to use when designing the Physical layer:

- Before importing metadata from a data warehouse into the Physical layer of your repository, eliminate all outer joins in the data warehouse. Use Extract, Transform, and Load (ETL) lookups to find them. Eliminating outer joins results in a more consistent record set, a simplified business model, and improved performance.
- To reduce problems that might be caused by importing physical joins, import the physical data without foreign keys and then create the keys as needed.

- You will probably import some tables into the Physical layer that you might not use right away but that you do not want to delete. One way to identify tables that you use in the Business Model and Mapping layer, is to assign aliases to physical tables before mapping them to the business model layer.

NOTE: To see the name of a table to which you assigned an alias, make sure you turn on the following option in the Tools > Options > General menu path: Display original names for alias in diagrams.

- Use a naming standard in the Physical Layer that identifies the logical components followed by the physical table name as shown in the following example:

Created By Employee (W_PERSON_D)

NOTE: In Version 7.7, the Physical layer object name is included in the Physical SQL. By using qualified Physical Layer names, debugging is made easier.

- An *opaque view* (a physical layer table that consists of a Select statement) should be used only if there is no other solution. Ideally, a physical table should be created, or alternatively materialized view. A traditional database view is not needed because it is identical to the opaque view.

Design Tips for the Business Model and Mapping Layer

The Business Model and Mapping layer organizes information by business model. Each business model contains logical tables. Logical tables are composed of logical columns. Logical tables have relationships to each other expressed by logical joins. The relationship between logical columns can be hierarchical, as defined in business model hierarchies. Logical tables map to the source data in the Physical layer. The mapping can include complex transformations and formulas.

The Business Model and Mapping layer defines the meaning and content of each physical source in business model terms. The Analytics Server uses these definitions to assign the appropriate sources for each data request.

You can change the names of physical objects independently from corresponding business model object names and properties, and vice versa. Any changes in the underlying physical databases or the mappings between the business model tables and physical tables might not change the view in the end-user applications that view and query the repository.

The logical schema defined in each business model needs to contain at least two logical tables. Relationships need to be defined between all the logical tables. For more information about business model schemas, see [“Process of Analytics Repository Planning and Design” on page 41](#). For more information about setting up the Business Model and Mapping layer, see [Chapter 5, “Creating and Administering the Business Model and Mapping Layer in a Repository.”](#)

The following is a list of tips to use when designing the Business Model and Mapping layer:

- Physically join fact extension tables to their base tables using one-to-many foreign key joins, and include them in the existing source for the logical table.
- Create new fact tables to support requirements when existing fact tables do not adequately meet the dimensional needs. Do not alter an existing fact table because it was designed to support other requirements.

- Create new fact tables or use the _FX to physically store links to other dimensions when they are not in the existing data model.
- Create the business model with one-to-many complex joins between logical dimension tables and the fact tables wherever possible. The business model should ideally resemble a simple star schema in which each fact table is linked directly to its dimensions.

NOTE: If you use the snowflake model, you might have more flexibility (for example, the ability to use outer joins) but it may create more columns in the presentation layer. However, it is recommended that you minimize the use of snowflake schemas.

- Join to _WID values instead of codes or names. Avoid Coded Records where the meaning of a record or field changes depending upon the value of a field. An example is to create joins to the W_LOV_D table on Code and Type, not with the ROW_WID as is currently done in the Siebel applications.
- Outer joins should be avoided in the reporting SQL. They can be eliminated in the ETL via a variety of techniques, but by doing so, not only can an additional table be removed from the report SQL, but the performance will also improve.
- Combine all similar dimensional attributes into one logical dimension table. Where needed, include data from other dimension tables into the main dimension source using aliases in the Physical layer tables. This should occur during the ETL process for optimal performance.
- Every logical dimension table should have a dimensional hierarchy associated with it. Make sure that all appropriate fact sources link to the proper level in the hierarchy using aggregation content. You set up aggregation content in the Content tab of the Logical Table Source properties window.
- Aggregate sources should be created as a separate Source in a logical table. For fact aggregates, use the Content tab of the Logical Table Source properties window to assign the correct logical level to each dimension.
- It is recommended that you use table aliases frequently to eliminate extraneous joins, including the following:
 - Eliminate all physical joins that cross dimensions (inter-dimensional circular joins) by using aliases.
 - Eliminate all circular joins (intra-dimensional circular joins) in a logical table source in the Physical Model by creating physical table aliases.
- Renaming an element in the Business Model and Mapping layer will automatically create an alias.
- Physically model dimension extension tables to their base tables using one-to-many foreign key joins, and included them in the existing source for the logical table.

You might want to create a source for just the Dimension _DX table and then create a one-to-many physical join between it and the fact tables to which it applies. This may help some queries perform better in cases where the _D table is not needed in the query.

NOTE: The physical key for both the dimension base table and the dimension extension table are identical, and the relationship is required to be 1:1 (one-to-one). Thus, although a circular join will occur in certain instances, it does not alter the record set or negatively impact performance.

- To prevent problems with aggregate navigation, make sure that each logical level of a dimension hierarchy contains the correct value in the field named Number of elements at this level. Fact sources are selected on a combination of the fields selected as well as the levels in the dimensions to which they link. By adjusting these values, you can alter the fact source selected by Siebel Business Analytics.
- Outer joins in logical table sources are always included in a query, even if the table source is not used. If possible, create one logical table source without the outer join and another with the outer join. Order the logical table source with the outer join after the non-outer join so that it will be used only when necessary.

Design Tips For the Presentation Layer

You set up the user view of a business model in the Presentation layer. The names of folders and columns in the Presentation layer appear in localized language translations. The Presentation layer is the appropriate layer in which to set user permissions. In this layer, you can do the following:

- You can show fewer columns than exist in the Business Model and Mapping layer. For example, you can exclude the key columns because they have no business meaning.
- You can organize columns using a different structure from the table structure in the Business Model and Mapping layer.
- You can display column names that are different from the column names in the Business Model and Mapping layer.
- You can set permissions to grant or deny users access to individual catalogs, tables, and columns.
- You can export logical keys to ODBC-based query and reporting tools.

For more information about setting up the Presentation layer, see [Chapter 6, "Creating and Maintaining the Presentation Layer in a Repository."](#)

The following is a list of tips to use when designing the Presentation layer:

- **Column aliases.** It is recommended that you do not use aliases for Presentation layer columns.
NOTE: Renaming an element in the Presentation layer will automatically create an alias. This prevents reports that reference the original element from failing.
- **Single table model.** For the greatest simplicity, you could construct a subject area that consists of a single table. To create a single table model, you first create a logical dimensional model, and then present it as a single table schema in the Administration Tool's Presentation layer. The logical dimensional model is required to set up the necessary metadata for the Analytics Server to navigate to the proper physical tables. For information about the Presentation layer, see ["Creating and Maintaining the Presentation Layer in a Repository" on page 135.](#)

Creating a New Analytics Repository File

Now that you have completed your planning and design tasks, you can create a repository. The first step in creating a repository is creating a repository file. Each time you save the repository, a dialog box asks if you want to check global consistency. You have the following options:

- **Yes.** Checks global consistency and then saves the repository file.
- **No.** Does not check global consistency and then saves the repository file.
- **Cancel.** Does not check global consistency and does not save the repository file.

NOTE: In offline editing, remember to save your repository from time to time. You can save a repository in offline mode even though the business models may be inconsistent.

To create a new repository file

- 1 From the Administration Tool menu, select File > New or click the New button in the toolbar.
- 2 Type a name for the repository.

A RPD file extension will automatically be added if you do not explicitly specify it. The default location for all repositories is in the Repository subdirectory, located in the Siebel Business Analytics software installation folder (\Siebel Analytics).

The new repository is empty. The remaining steps in the repository setup process, as outlined in [“Roadmap for Planning and Setting Up an Analytics Repository” on page 41](#), describe the steps you should follow to set up and populate repositories.

NOTE: When populating the repository, make sure that no repository objects in any layer are named Administrator. This name is reserved for the user ID of the Analytics Server administrator.

After Creating a Repository File

After you create a repository file in the Administration Tool, you can import tables into the Physical layer.

NOTE: The typical order is to create the Physical layer first, the Business Model and Mapping layer next, and the Presentation layer last. However, you can work on each layer at any stage in creating a repository. You can set up security when you are ready to begin testing the repository.

For more information, see [“Creating and Administering the Physical Layer in a Repository” on page 57](#).

4

Creating and Administering the Physical Layer in a Repository

This section is part of the roadmap for planning and setting up a repository. For more information, see [“Planning and Creating a Siebel Business Analytics Repository” on page 41](#).

Before you create the physical layer of a repository, you need to plan, design, and then create a repository file. For more information, see [“Process of Analytics Repository Planning and Design” on page 41](#)

The Physical layer of the Administration Tool defines the data sources to which the Analytics Server submits queries and the relationships between physical databases and other data sources that are used to process multiple data source queries.

NOTE: If your organization has licensed Oracle’s Siebel Analytics Platform Server Extended product or Oracle’s Siebel Analytics Stand-Alone product, you are authorized to add databases and connection pools to the physical layer. If your organization has licensed a different product, you can only use the databases or connection pools that are provided with the product.

The first step in creating the physical layer is to create the schema. You can import the physical schema for supported data source types. You can import schemas or portions of schemas from existing data sources. Additionally, you can create the physical layer manually.

CAUTION: It is strongly recommended that you import the physical schema.

This section provides the following topics to help you use the Administration Tool to create the physical layer of a repository:

- [Process of Creating the Physical Layer from Relational Data Sources on page 58](#)
- [Process of Creating the Physical Layer from Multidimensional Data Sources on page 60](#)
- [Setting Up Database Objects on page 62](#)
- [Setting Up Connection Pools on page 65](#)
- [About Physical Tables on page 81](#)
- [Creating and Setting Up Physical Tables on page 83](#)
- [Creating Physical Layer Folders on page 92](#)
- [About Physical Joins on page 95](#)
- [Defining Physical Foreign Keys and Joins on page 96](#)
- [Using Database Hints on page 99](#)

Process of Creating the Physical Layer from Relational Data Sources

Importing the physical schema saves you time and effort by importing the structure for the physical layer. You can import schema for supported data sources. If you do not import the schema, you must create each table, primary key, foreign key, and any other objects against which you want to generate queries. Data from these sources can be displayed on Oracle's Siebel Intelligence Dashboards.

The following is a list of tips to help you when importing a physical schema:

- When you import schema for most databases, the default is to import tables, primary keys, and foreign keys.

NOTE: It is recommended that you not import foreign keys from a database because the process is lengthy when importing a large number of tables.

- When you import physical tables, be careful to limit the import to only those tables that contain data that are likely to be used in the business models you create. You can use a filter (table mask) to limit the number of tables that appear in the import list. This makes it easier to locate and select the tables that you want to import.
- You can also import database views, aliases, synonyms, and system tables. Import these objects only if you want the Analytics Server to generate queries against them.
- Importing large numbers of extraneous tables and other objects adds unnecessary complexity and increases the size of the repository.

To create the physical layer by importing the schema, complete the following tasks in the sequence shown:

- [Importing a Physical Schema from Relational Data Sources on page 58](#)
- [Setting Up Database Objects on page 62](#)
- [Setting Up Connection Pools on page 65](#)
- [About Physical Tables on page 81](#)
- [Creating and Setting Up Physical Tables on page 83](#)

Importing a Physical Schema from Relational Data Sources

This topic is part of the ["Process of Creating the Physical Layer from Relational Data Sources" on page 58](#).

You can import physical schema for supported data source types. This task is part of ["Process of Creating the Physical Layer from Relational Data Sources" on page 58](#).

When you can use importing to create the physical schema, you need to select one of the following import options and the appropriate connection type:

- **From a database.** Available in Offline and Online modes. Use this option when you have all database connections set up on your machine. You can use the following connection types with the Import option:
 - Most physical schema imports are performed using an ODBC connection type.
 - Native database gateways for schema import are supported for only DB2 (using DB2 CLI or DB2 CLI Unicode) and XML connection types. For more information about importing XML data using the Analytics Server XML gateway, see [“Analytics Server XML Gateway Example” on page 350](#).
 - You can use a table mask to limit (filter) the list of tables that appear in the Import dialog box. When you have one or more specific tables that you want to import, using a table mask helps locate the tables.
- **From a repository.** Available in Offline and Online modes. The Repository Import Wizard takes you through this process.
- **Import through the Analytics Server.** Available in Online mode. Use this option when you want to use the Analytics Server connections to import schema. This feature allows administrators to use the DSNs of the Analytics Server machine to import physical schema information. Connectivity software and duplicate DSNs do not have to reside on the administrator's machine and the Analytics Server machine. You can use the following connection types with the Import option:
 - Available connection types are ODBC, DB2 CLI, DB2 CLI (Unicode), and XML.
 - When it is running on a UNIX platform, the Analytics Server does not support importing schema using an ODBC connection type.

To import a physical schema from an ODBC connection type

- 1 In the Administration Tool, select File > Import, and then select the source type of your physical schema from the following options:
 - from Database
 - through Server
 - from Repository
- 2 In the Select Data Source dialog box, perform the following steps:
 - a From the Connection Type drop-down list, select the correct type.

NOTE: It is recommended that you use ODBC 3.5 or DB2 CLI Unicode for importing schema with International characters such as Japanese table and column names.
 - b In the DSN list, select a data source from which to import the schema.

When you import through the Analytics Server, the DSN entries are on the Analytics Server, not on the local machine.
 - c Click OK.

- 3 If a logon screen appears, type a valid user ID and password for the data source, and then click OK.

The administrator must supply the user name and password for the data source. If you are performing an import on the administrator's machine, the user name will be obtained from the machine's DSN configuration.

- 4 In the Import dialog box, select the check boxes for the types of objects that you want to import. For example, Tables, Keys, and Foreign Keys.

Some objects check boxes are automatically selected. These default selections depend on your data source.

- 5 To use a table mask, type a table mask such as F%, and then click the highest level database folder.

Tables that meet the filter criteria appear.

- 6 Select the tables and columns you want to import.

- 7 After you select all the objects you want to import, click Import or drag and drop the objects into the Physical layer.

Process of Creating the Physical Layer from Multidimensional Data Sources

This section describes how you use the Administration Tool to add a multidimensional data source to the physical layer of a repository. The ability to use multidimensional data sources allows the Analytics Server to connect to sources such as Microsoft Analysis Services and extract data. Data from these sources can be displayed on Siebel Intelligence Dashboards.

Siebel Business Analytics connects to the multidimensional source using XML for Analysis (XMLA) standards protocol. This requires that the target data source have a fully functional Web services interface available. The standard dictates the various protocols that the Analytics Server can use to connect to the target and query data.

Importing data from a multidimensional source creates the metadata of the data source in the analytics repository. The primary differences between setting up multidimensional data sources and relational data sources are in the physical layer. The setup in the business model and presentation layers for multidimensional data sources and relational data sources is almost identical.

To create a physical layer for a multidimensional data source, complete the following tasks in the sequence shown:

- [Importing a Physical Schema from Multidimensional Data Sources on page 61](#)
- [Setting Up Database Objects on page 62](#)
- [Setting Up Connection Pools on page 65](#)
- [About Physical Tables on page 81](#)
- [Creating and Setting Up Physical Tables on page 83](#)

Importing a Physical Schema from Multidimensional Data Sources

This topic is part of [“Process of Creating the Physical Layer from Multidimensional Data Sources” on page 60.](#)

CAUTION: Although you can create a physical schema from a multidimensional data source, it is strongly recommended that you import it.

Siebel Business Analytics uses XMLA standards to import data from a multidimensional data source into the analytics repository. During the import process, each cube in a multidimensional data source is created as a single physical cube table. Siebel Business Analytics imports the cube, including its metrics, dimensions and hierarchies. After importing the cubes, you need to make sure that the physical cube columns have the correct aggregation rule and that the default member type ALL is correctly imported for a hierarchy. For more information, see [“Adding a Hierarchy to a Physical Cube Table” on page 89.](#)

The following list includes information that you should consider before importing data:

- Microsoft Analysis Services is the only supported XMLA-compliant data source currently available.
- Siebel Business Analytics only imports the dimensions and hierarchies that are supported by Siebel Business Analytics. Therefore, if a cube has a ragged hierarchy or a parent-child hierarchy, it will not be imported.
- It is recommended that you remove hierarchies and columns from the physical layer if they will not be used in the business model. This eliminates maintaining unnecessary objects in the Administration Tool and might result in better performance.

To import a physical schema from multidimensional data sources

- 1 In the Administration Tool, select File > Import from XMLA.
- 2 In the Import From XMLA dialog box, in the URL field, type the URL for the Web service that acts as the XMLA provider.

Obtain the URL connection string from your Microsoft Analysis Services Administrator.
- 3 In the username and password fields, type a valid username and password for the data source.

The administrator must supply the user name and password for the data source.
- 4 Click OK.

The Administration Tool connects to the destination source to obtain a list of the available data catalogs (databases).
- 5 In the second Import From XMLA dialog box, expand the data source and catalog (database), if necessary, and select the catalogs (databases) or cubes to import.

For more information The OK button is unavailable until you select a downloadable object.
- 6 After you select all the objects you want to import, click OK.

Setting Up Database Objects

This topic is part of the [“Process of Creating the Physical Layer from Relational Data Sources”](#) on page 58 and the [“Process of Creating the Physical Layer from Multidimensional Data Sources”](#) on page 60.

Importing a schema automatically creates a database object for the schema but you need to set up the database properties.

For more information about supported databases, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

To create or edit database objects in the physical layer, perform the following tasks:

- [About Database Types in the Physical Layer on page 62](#)
- [Creating a Database Object Manually in the Physical Layer on page 62](#)
- [Specifying SQL Features Supported by a Database on page 64](#)

About Database Types in the Physical Layer

If you import the physical schema into the Physical layer, the database type is usually assigned automatically. The following list contains additional information about automatic assignment of database types:

- **Relational data sources.** During the import process, some ODBC drivers provide the Analytics Server with the database type. However, if the server cannot determine the database type, an approximate ODBC type is assigned to the database object. Replace the ODBC type with the closest matching entry from the database type drop-down menu.
- **Multidimensional data sources.** Microsoft Analysis Services is the only supported XMLA-compliant data source currently available. Therefore, when you import a multidimensional data source, Analysis Services 2000 is automatically assigned as the database type and cannot be changed.

Creating a Database Object Manually in the Physical Layer

If you create a database object manually, you need to manually set up all database elements such as connection pool, tables, and columns.

NOTE: For multidimensional data sources, if you create the physical schema in the physical layer of the repository, you need to create one database in the physical layer for each cube, or set of cubes, that belong to the same catalog (database) in the data source. A physical database can have more than one cube. However, all of these cubes must be in the same catalog in the data source. You specify a catalog in the Connection Pool dialog box.

CAUTION: It is strongly recommended that you import your physical schema.

To create a database object

- 1 In the Physical layer of the Administration Tool, right-click and choose New Database.
Make sure that no object is selected when you right-click.
- 2 In the Database dialog box, in the General tab, complete the fields using [Table 8 on page 63](#) as a guide.

Table 8. Fields General Tab of the Database Dialog Box

Field	Description
Database	The database type for your database. For more information about using the Features tab to examine the SQL features supported by the specified database type, see “Specifying SQL Features Supported by a Database” on page 64 .
Persist Connection Pool	To use a persistent connection pool, you must set up a temporary table first. For more information, see “Setting Up the Persist Connection Pool Property” on page 79 .
Allow populate queries by default (check box)	When checked, allows everyone to execute POPULATE SQL. If you want most but not all users to be able to execute POPULATE SQL, check this option and then limit queries for specific users or groups. For more information about limiting queries, see “Managing Query Execution Privileges” on page 343 .
Allow direct database requests by default (check box)	<p>When checked, allows everyone to execute physical queries. Analytics Server will send unprocessed, user-entered, physical SQL directly to an underlying database. The returned results set can be rendered in the Siebel Business Analytics Web user interface, and then charted, rendered in a dashboard, and treated as an analytics request.</p> <p>If you want most but not all users to be able to execute physical queries, check this option and then limit queries for specific users or groups. For more information about limiting queries, see “Managing Query Execution Privileges” on page 343.</p> <p>CAUTION: If configured incorrectly, this can expose sensitive data to an unintended audience. For more information, see “Recommendations for Allowing Direct Database Requests by Default” on page 63.</p> <p>For more information about executing physical SQL, see <i>Siebel Business Analytics User Guide</i>.</p>

Recommendations for Allowing Direct Database Requests by Default

This property allows all users to execute physical queries. If configured incorrectly, it can expose sensitive data to an unintended audience. Use the following recommended guidelines when setting this database property:

- The Analytics Server should be configured to accept connection requests only from a machine on which the Analytics Server, Siebel Business Analytics Web Server, or Oracle's Siebel Scheduler Server are running. This restriction should be established at the TCP/IP level using the Siebel Business Analytics Web Server IP address. This will allow only a TCP/IP connection from the IP Address of the Siebel Business Analytics Web Server.
 - To prevent users from running NQCMD by logging in remotely to this machine, you should disallow access by the following to the machine where you installed Siebel Business Analytics Web:
 - TELNET
 - Remote shells
 - Remote desktops
 - Teleconferencing software (such as Windows NetMeeting)
- If necessary, you might want to make an exception for users with Administrator permissions.
- Only users with Administrator permissions should be allowed to perform the following tasks:
 - TELNET into the Analytics Server and Siebel Business Analytics Web Server machines to perform tasks such as running NQCMD for cache seeding.
 - Access to the advanced SQL page of Answers to create reports. For more information, see *Siebel Business Analytics Web Administration Guide*.
 - Setup group/user-based permissions on the Siebel Business Analytics Web Server to control access to editing (preconfigured to allow access by Web Administrators) and executing (preconfigured to not allow access by anyone) direct database requests. For more information, see *Siebel Business Analytics Web Administration Guide*.

Specifying SQL Features Supported by a Database

When you import the schema or specify a database type in the General tab of the Database dialog box, the Feature table is automatically populated with default values appropriate for the database type. These are the SQL features that the Analytics Server uses with this data source.

You can tailor the default query features for a database. For example, if a data source supports left outer join queries but you want to prohibit the Analytics Server from sending such queries to a particular database, you can change this default setting in the Feature table.

CAUTION: Be very careful when modifying the Feature table. If you enable SQL features that the database does not support, your query may return errors and unexpected results. If you disable supported SQL features, the server could issue less efficient SQL to the database.

To specify SQL features supported by a database

- 1 In the Physical layer of the Administration Tool, double-click the database.
- 2 In the Database dialog box, click the Features tab.

- 3 In the Features tab, use the information in [Table 9 on page 65](#) to help you specify SQL features.

Table 9. SQL Feature Tab Descriptions

Field or Button	Description
Value	A check box that allows you to specify additional SQL features. Select to enable a query type or clear to disable a query type. It is strongly recommended that you do not disable the default SQL features.
Default	A check box that identifies default SQL features. Default SQL features that are supported by the database type of the data source are automatically selected.
Ask DBMS	A button that is used only if you are installing and querying a database that has no Features table. It allows you to query this database for Feature table entries. For more information, see "Changing Feature Table Entries Using Ask DBMS" on page 65 .
Revert to Defaults	A button that restores the default values.
Find	A button that allows you to type a string to help you locate a feature in the list.
Find Again	A button that becomes available after you click Find. Allows you to perform multiple searches for the same string.

Changing Feature Table Entries Using Ask DBMS

You should use the Ask DBMS button only if installing and querying a database that has no Features table.

NOTE: The Ask DBMS button is not available if you are using an XML or a multidimensional data source.

The Ask DBMS button on the Features tab of the Database dialog box allows you to query a database for the SQL features it supports. You can change the entries that appear in the Feature table based on your query results.

CAUTION: Be very careful when using Ask DBMS. The results of the features query are not always an accurate reflection of the SQL features actually supported by the data source. You should only use this functionality with the assistance of Oracle Technical Support.

Setting Up Connection Pools

This topic is part of the ["Process of Creating the Physical Layer from Relational Data Sources" on page 58](#) and the ["Process of Creating the Physical Layer from Multidimensional Data Sources" on page 60](#).

The *connection pool* is an object in the Physical layer that describes access to the data source. It contains information about the connection between the Analytics Server and that data source.

The Physical layer in the Administration Tool contains at least one connection pool for each database. When you create the physical layer by importing a schema for a data source, the connection pool is created automatically. You can configure multiple connection pools for a database. Connection pools allow multiple concurrent data source requests (queries) to share a single database connection, reducing the overhead of connecting to a database.

NOTE: It is recommended that you create a dedicated connection pool for initialization blocks. For additional information, see [“Creating or Changing Connection Pools” on page 68](#).

For each connection pool, you must specify the maximum number of concurrent connections allowed. After this limit is reached, the Analytics Server routes all other connection requests to another connection pool or, if no other connection pools exist, the connection request waits until a connection becomes available.

Increasing the allowed number of concurrent connections can potentially increase the load on the underlying database accessed by the connection pool. Test and consult with your DBA to make sure the data source can handle the number of connections specified in the connection pool. Also, if the data sources have a charge back system based on the number of connections, you might want to limit the number of concurrent connections to keep the charge-back costs down.

In addition to the potential load and costs associated with the database resources, the Analytics Server allocates shared memory for each connection upon server startup. This raises the number of connections and increases Analytics Server memory usage.

[Table 10 on page 66](#) contains a brief description of preconfigured connection pools.

Table 10. Preconfigured Connection Pools

Connection Name	Description
BBB Data Warehouse	Enterprise visibility Oracle's Siebel Data Warehouse. Siebel operational applications Siebel Business Analytics customers only.
BBB OLTP	Enterprise visibility Oracle's Siebel transactional database. Oracle's Siebel operational applications (Siebel Business Analytics customers only).
BBB XLS Data	DSN is the same as the connection pool name. Data is stored in C:\Data\Applications\BBB\BBB. Oracle's Siebel operational applications (Siebel Business Analytics customers only).
DataMining Demo	Data Warehouse visibility. Oracle's Siebel Data Mining customers only.
ERM Siebel OLTP	Oracle's Siebel Workforce Analytics connection to Siebel transactional database. Workforce Siebel Business Analytics customers only.

Table 10. Preconfigured Connection Pools

Connection Name	Description
Externalized Metadata Strings	<p>Connection to the Siebel operational application database to load the translations of Metadata Strings. Financial Services customers and all customers deploying Siebel Business Analytics in a language other than English.</p> <p>NOTE: This connection pool is configured to be the same as the Siebel transactional database.</p>
Forecasting Siebel Data Warehouse	<p>Connection to the Siebel Data Warehouse. Forecasting Siebel Business Analytics customers only.</p>
Forecasting Siebel OLTP	<p>Connection to the Siebel transactional database. Real-time Forecasting Siebel Business Analytics customers only.</p>
Incentive Compensation Siebel OLTP	<p>Database connection to the Siebel transactional database. Incentive Compensation Siebel Business Analytics customers only.</p>
Pharma Data Warehouse	<p>Connection to the Pharmaceutical data warehouse. Pharmaceutical industry-specific customers only.</p>
Real-time OLTP	<p>Connection to the Siebel transactional database. Real-time Siebel Business Analytics (all customers).</p>
SIA Data Warehouse	<p>Connection for Oracle's Siebel Industry Applications data warehouse. For Siebel Industry Applications customers only.</p>
Siebel Analytics Usage	<p>Usage Tracking Writer Connection Pool connects to the database where you store the usage statistics of the Analytics Server. Optional for all customers.</p> <p>NOTE: Using this feature requires that Siebel Analytics Scheduler Server be set up to load the usage statistics into the database.</p>
Siebel Data Warehouse	<p>Database connection to the Siebel Data Warehouse. Siebel Business Analytics applications customers only.</p>
Siebel OLTP	<p>NOTE: There are two connection pools to the Siebel transactional database. Both should be configured properly.</p> <p>OLTP DbAuth connects to the Siebel transactional database for Authentication and Authorization. The user name and password is preconfigured to USER and PASSWORD and should be left as such if you want to use database logons to authenticate users.</p> <p>All customers.</p>

Table 10. Preconfigured Connection Pools

Connection Name	Description
UQ Siebel OLTP	Connection to the Siebel transactional database. Oracle's Siebel Universal Queuing customers only.
Usage Accelerator Datawarehouse	Database connection to the Siebel Data Warehouse. Siebel Business Analytics applications customers only.

This section includes the following topics:

- [Creating or Changing Connection Pools on page 68](#)
- [Setting Up Connection Pool Properties for Multidimensional Data Sources on page 75](#)
- [Setting Up Additional Connection Pool Properties for an XML Data Source on page 77](#)
- [Setting Up the Persist Connection Pool Property on page 79](#)

Creating or Changing Connection Pools

You must create a database object before you create a connection pool. Typically, the database object and connection pool are created automatically when you import the physical schema. You create or change a connection pool in the Physical layer of the Administration Tool.

CAUTION: It is strongly recommend that customers use OCI for connecting to Oracle. ODBC should only be used to import from Oracle.

This section contains the following topics:

- [About Connection Pools for Initialization Blocks on page 68](#)
- [Setting Up General Properties For Connection Pools on page 69](#)
- [Setting Up Write-Back Properties on page 73](#)
- [Setting Up Connection Pool Properties for Multidimensional Data Sources on page 75.](#)

About Connection Pools for Initialization Blocks

It is recommended that you create a dedicated connection pool for initialization blocks. This connection pool should not be used for queries.

Additionally, it is recommended that you isolate the connections pools for different types of initialization blocks. This also makes sure that authentication and login-specific initialization blocks do not slow down the login process. The following types should have separate connection pools:

- All Authentication and login-specific initialization blocks such as language, externalized strings, and group assignments.
- All initialization blocks that set session variables.

- All initialization blocks that set repository variables. These initialization blocks should always be run using the system administrator user login.

Be aware of the number of these initialization blocks, their scheduled refresh rate, and when they are scheduled to run. Typically, it would take an extreme case for this scenario to affect resources. For example, refresh rates set in minutes, greater than 15 initialization blocks that refresh concurrently, and a situation in which either of these scenarios could occur during prime user access time frames. To avoid straining the available resources, you might want to disable query logging for the default Analytics Server administrator.

Initialization blocks should be designed so that the maximum number of Analytics Server variables may be assigned by each block. For example, if you have five variables, it is more efficient and less resource intensive to construct a single initialization block containing all five variables. When using one initialization block, the values will be resolved with one call to the back end tables using the initialization string. Constructing five initialization blocks, one for each variable, would result in five calls to the back end tables for assignment.

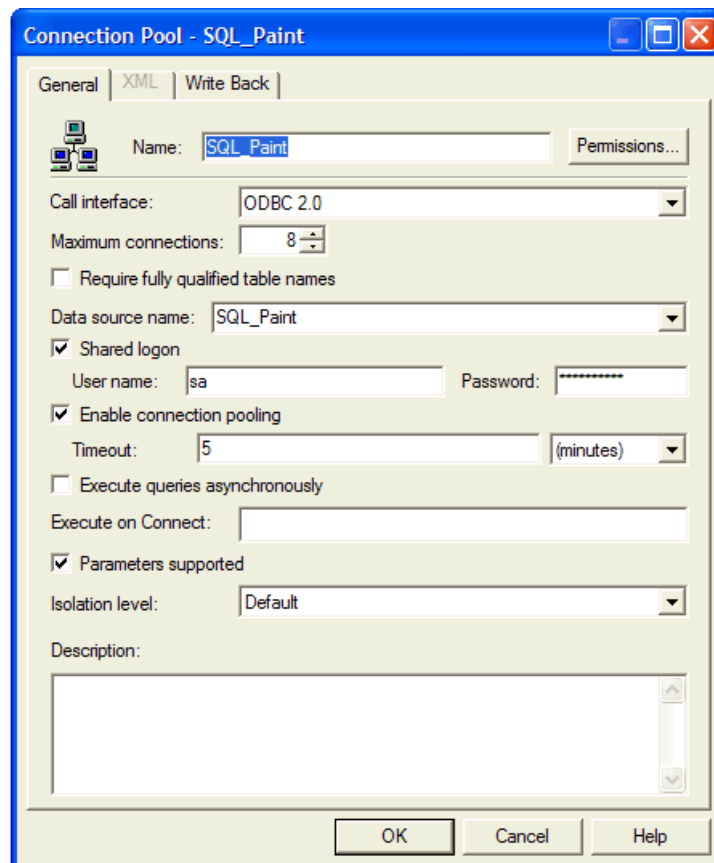
Setting Up General Properties For Connection Pools

Use this section to complete the General tab.

To set up general properties for connection pools

- 1 In the Physical layer of the Administration Tool, right-click a database and choose New Object > Connection Pool, or double-click an existing connection pool.

The following is an illustration of the General tab in the Connection Pool dialog box.



- 2 In the Connection Pool dialog box, click the General tab, and then complete the fields using information in [Table 11 on page 71](#).

Properties that are specific to a multidimensional data sources can be found in [Table 13 on page 77](#).

Table 11. Connection Pool General Properties

Field or Button	Description
Call interface	The application program interface (API) with which to access the data source. Some databases may be accessed using native APIs, some use ODBC, and some work both ways. If the call interface is XML, the XML tab is available but options that do not apply to XML data sources are not available.
Data source name	The drop-down list shows the User and System DSNs configured on your system. A data source name that is configured to access the database to which you want to connect. The data source name needs to contain valid logon information for a data source. If the information is invalid, the database logon specified in the DSN will fail.
Enable connection pooling	Allows a single database connection to remain open for the specified time for use by future query requests. Connection pooling saves the overhead of opening and closing a new connection for every query. If you do not select this option, each query sent to the database opens a new connection.
Execute on Connect	Allows the Analytics Server administrator to specify a command to be executed each time a connection is made to the database. The command may be any command accepted by the database. For example, it could be used to turn on quoted identifiers. In a mainframe environment, it could be used to set the secondary authorization ID when connecting to DB2 to force a security exit to a mainframe security package such as RACF. This allows mainframe environments to maintain security in one central location.
Execute queries asynchronously	Indicates whether the data source supports asynchronous queries.

Table 11. Connection Pool General Properties

Field or Button	Description
Isolation level	<p>For ODBC and DB2 gateways, the value sets the transaction isolation level on each connection to the back-end database. The isolation level setting controls the default transaction locking behavior for all statements issued by a connection. Only one of the options can be set at a time. It remains set for that connection until it is explicitly changed.</p> <p>The following is a list of the options:</p> <p>Committed Read. Specifies that shared locks are held while the data is read to avoid dirty reads. However, the data can be changed before the end of the transaction, resulting in non repeatable reads or phantom data.</p> <p>Dirty Read. Implements dirty read (isolation level 0 locking). When this option is set, it is possible to read uncommitted or dirty data, change values in the data, and see rows appear or disappear in the data set before the end of the transaction. This is the least restrictive of the isolation levels.</p> <p>Repeatable Read. Places locks on all data that is used in a query, preventing other users from updating the data. However, new phantom rows can be inserted into the data set by another user and are included in later reads in the current transaction.</p> <p>Serializable. Places a range lock on the data set, preventing other users from updating or inserting rows into the data set until the transaction is complete. This is the most restrictive of the four isolation levels. Because concurrency is lower, use this option only if necessary.</p>
Maximum connections	<p>The maximum number of connections allowed for this connection pool. The default is 10. This value should be determined by the database make and model and the configuration of the hardware box on which the database runs as well as the number of concurrent users who require access.</p> <p>NOTE: For deployments with Intelligence Dashboard pages, consider estimating this value at 10% to 20% of the number of simultaneous users multiplied by the number of requests on a dashboard. This number may be adjusted based on usage. The total number of all connections in the repository should be less than 800. To estimate the maximum connections needed for a connection pool dedicated to an initialization block, you might use the number of users concurrently logged on during initialization block execution.</p>
Name	<p>The name for the connection pool. If you do not type a name, the Administration Tool generates a name. For multidimensional and XML data sources, this is prefilled.</p>
Parameters supported	<p>If the database features table supports parameters and the connection pool check box property for parameter support is unchecked, then special code executes that allows the Analytics Server to push filters (or calculations) with parameters to the database. The Analytics Server does this by simulating parameter support within the gateway/adaptor layer by sending extra SQLPrepare calls to the database.</p>

Table 11. Connection Pool General Properties

Field or Button	Description
Permissions	Assigns permissions for individual users or groups to access the connection pool. You can also set up a privileged group of users to have its own connection pool.
Require fully qualified table names	<p>Select this check box, if the database requires it.</p> <p>When this option is selected, all requests sent from the connection pool use fully qualified names to query the underlying database. The fully qualified names are based on the physical object names in the repository. If you are querying the same tables from which the physical layer metadata was imported, you can safely check the option. If you have migrated your repository from one physical database to another physical database that has different database and schema names, the fully qualified names would be invalid in the newly migrated database. In this case, if you do not select this option, the queries will succeed against the new database objects.</p> <p>For some data sources, fully qualified names might be safer because they guarantee that the queries are directed to the desired tables in the desired database. For example, if the RDBMS supports a master database concept, a query against a table named <i>foo</i> first looks for that table in the master database, and then looks for it in the specified database. If the table named <i>foo</i> exists in the master database, that table is queried, not the table named <i>foo</i> in the specified database.</p>
Shared logon	<p>Select the Shared logon check box if you want all users whose queries use the connection pool to access the underlying database using the same user name and password.</p> <p>If this option is selected, then all connections to the database that use the connection pool will use the user name and password specified in the connection pool, even if the user has specified a database user name and password in the DSN (or in the Siebel user configuration).</p> <p>If this option is not selected, connections through the connection pool use the database user ID and password specified in the DSN or in the Siebel user profile.</p>
Timeout (Minutes)	<p>Specifies the amount of time, in minutes, that a connection to the data source will remain open after a request completes. During this time, new requests use this connection rather than open a new one (up to the number specified for the maximum connections). The time is reset after each completed connection request.</p> <p>If you set the timeout to 0, connection pooling is disabled; that is, each connection to the data source terminates immediately when the request completes. Any new connections either use another connection pool or open a new connection.</p>

Setting Up Write-Back Properties

Use this section to complete the Write Back tab in the Connection Pool dialog box.

To set up write-back properties for connection pools

- 1 In the Physical layer of the Administration Tool, right-click a database and select New Object > Connection Pool, or double-click an existing connection pool.
- 2 In the Connection Pool dialog box, click the Write Back tab.
- 3 In the Write Back tab, complete the fields using [Table 12 on page 74](#) as a guide.

Table 12. Field Descriptions for Write Back Tab

Field	Description
Owner	Table owner name used to qualify a temporary table name in a SQL statement, for example to create the table owner.tablename. If left blank, the user name specified in the writeable connection pool is used to qualify the table name and the Shared Logon field on the General tab should also be set.
Prefix	When the Analytics Server creates a temporary table, these are the first two characters in the temporary table name. The default value is TT.
Database Name	Database where the temporary table will be created. This property applies only to IBM OS/390 because IBM OS/390 requires database name qualifier to be part of the CREATE TABLE statement. If left blank, OS/390 will default the target database to a system database for which the users may not have Create Table privileges.
Tablespace Name	Tablespace where the temporary table will be created. This property applies to OS/390 only as OS/390 requires tablespace name qualifier to be part of the CREATE TABLE statement. If left blank, OS/390 will default the target database to a system database for which the users may not have Create Table privileges.
Bulk Insert Buffer Size (KB)	Used for limiting the number of bytes each time data is inserted in a database table.

Table 12. Field Descriptions for Write Back Tab

Field	Description
Bulk Insert	Controls the batch size for an insert in a database table.
Transaction Boundary	
Unicode Database Type	<p>Select this check box when working with columns of an explicit Unicode data type, such as NCHAR, in an Unicode database. This makes sure that the binding is correct and data will be inserted correctly. Different database vendors provide different character data types and different levels of Unicode support. Use the following general guidelines to determine when to set this check box:</p> <ul style="list-style-type: none"> ■ On a database where CHAR data type supports Unicode and there is no separate NCHAR data type, do not select this check box. ■ On a database where NCHAR data type is available, it is recommended to select this check box. ■ On a database where CHAR and NCHAR data type are configured to support Unicode, selecting this check box is optional. <p>NOTE: Unicode and non-Unicode datatypes cannot coexist a single non-Unicode database. For example, mixing the CHAR and NCHAR data types in a single non-Unicode database environment is not supported.</p>

Setting Up Connection Pool Properties for Multidimensional Data Sources

When you import an external multidimensional data source, the connection pool is automatically set up in the physical layer. [Table 13 on page 77](#) describes the properties in this dialog box that are unique to multidimensional data sources.

If you need to add a connection pool manually, use the following guidelines:

- Specify a valid URL.
- Specify a username and password, if required.
- Using the Get Catalogs button, choose a data source.
- For each data source chosen, browse and choose a catalog.

To set up connection pools for a multidimensional data source

- 1 In the Physical layer of the Administration Tool, right-click a database and select New Object > Connection Pool, or double-click an existing connection pool.

The following is an illustration of the General tab for a multidimensional data source in the Connection Pool dialog box.

The screenshot shows the 'Connection Pool - Connection Pool' dialog box with the 'General' tab selected. The dialog has three tabs: 'General', 'XML', and 'Write Back'. The 'General' tab contains the following fields and controls:

- Name:** A text box containing 'Connection Pool' and a 'Permissions...' button.
- Call interface:** A dropdown menu set to 'XMLA'.
- Maximum connections:** A numeric spinner box set to '10'.
- URL:** A text box containing 'http://FoodMart01/xmla/msxisapi.dll'.
- Data source information:**
 - Data source:** A dropdown menu set to 'Provider=MSOLAP;Data Source=local'.
 - Catalog:** A dropdown menu set to 'FoodMart 2000' and a 'Get Catalogs' button.
- Connection properties:**
 - ☒ **Shared logon**
 - User name:** A text box.
 - Password:** A text box with masked characters (asterisks).
 - URL loading time-out:** A numeric spinner box set to '5' and a '(minutes)' dropdown menu.
 - ☐ **Use session**
- Description:** A large text area.

At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Help'.

- 2 In the Connection Pool dialog box, in the General tab, complete the fields using information in Table 13 on page 77.

Table 13. Multidimensional Data Source Connection Pool General Properties

Property	Description
Data Source Information: Catalog	The list of catalogs available, if you imported data from your data source. The cube tables correspond to the catalog you use in the connection pool.
Data Source Information: Data Source	The vendor-specific information used to connect to the multidimensional data source. Consult your multidimensional data source administrator for setup instructions because specifications may change. For example, if you use v1.0 of the XML for Analysis SDK then the value should be Provider-MSOLAP;Data Source-Local. If using v1.1, then it should be Local Analysis Server.
Shared logon	<p>Select the Shared logon check box if you want all users whose queries use the connection pool to access the underlying database using the same user name and password.</p> <p>If this option is selected, then all connections to the database that use the connection pool will use the user name and password specified in the connection pool, even if the user has specified a database user name and password in the DSN (or in the Siebel user configuration).</p> <p>If this option is not selected, connections through the connection pool use the database user ID and password specified in the DSN or in the Siebel user profile.</p>
URL	<p>The URL to connect to the XMLA provider. It points to the XMLA virtual directory of the machine hosting the cube. This virtual directory needs to be associated with msxisapi.dll (part of the Microsoft XML for Analysis SDK installation). For example, the URL might look like the following:</p> <p>http://SDCDL360i101/xmla/msxisap.dll</p>
Use session	Controls whether queries go through a common session. Consult your multidimensional data source administrator to determine whether this option should be enabled. Default is Off (not checked).

Setting Up Additional Connection Pool Properties for an XML Data Source

Use the XML tab of the Connection Pool dialog box to set additional properties for an XML data source.

CAUTION: The XML tab of the Connection Pool dialog box provides the same functionality as the XML tab of the Physical Table dialog box. However, when you set the properties in the XML tab of the Physical Table dialog box you will override the corresponding settings in the Connection Pool dialog box.

To set up connection pool properties for an XML data source

- 1 Right-click the XML database, select New Object > Connection Pool.
- 2 In the Connection Pool dialog box, click the XML tab.
- 3 Complete the fields, using Table 14 as a guide.

Table 14. XML Connection Pool Properties

Property	Description
Connection method	Used for XML Server data source.
Search script	
Connection properties	Default is 10.
Maximum connections	
Connection properties	Used for XML data source. Time-out interval for queries. The default is 15 minutes.
URL loading time out	<p>If you specified a URL to access the data source, set the URL loading time-out as follows:</p> <ul style="list-style-type: none"> ■ Select a value from the drop-down list (Infinite, Days, Hours, Minutes or Seconds). ■ Specify a whole number as the numeric portion of the interval.
Connection properties	Used for XML data source. The refresh interval is analogous to setting cache persistence for database tables. The URL refresh interval is the time interval after which the XML data source will be queried again directly rather than using results in cache. The default setting is infinite, meaning the XML data source will never be refreshed.
URL refresh interval	<p>If you specified a URL to access the data source, set the URL refresh interval.</p> <ul style="list-style-type: none"> ■ Select a value from the drop-down list (Infinite, Days, Hours, Minutes or Seconds). ■ Specify a whole number as the numeric portion of the interval.
Query input supplements	Used for XML Server data source.
Header file/Trailer file	
Query output format	<p>Choose only XML for an XML data source.</p> <p>Other choices are available for an XML Server data source.</p>

Table 14. XML Connection Pool Properties

Property	Description
XPath expression	<p>An XPath expression is a simple XSLT transformation rule that fits into one line. It is not essential, given the support of XSLT, but it is supported for convenience. A sample entry might be <code>*/BOOK[not(PRICE>'200')]</code>.</p> <ul style="list-style-type: none"> ■ For an XML Server data source, you cannot specify an XPath expression on the XML tab of the Physical Table object.
XSLT file	<p>An XSLT file contains formatting rules written according to the XSLT standard. It defines how an XML file may be transformed. The current implementation of the XML gateway does not support all XML files, only those that resemble a tabular form, with repeating second level elements. To increase the versatility of the XML gateway, customers can specify an XSLT file to preprocess an XML file to a form that the Analytics Server supports. Specifying the XSLT file in the connection pool applies it to all the XML physical tables in the connection pool.</p> <ul style="list-style-type: none"> ■ For an XML data source, you can specify an XSLT file on a per-table basis on the XML tab of the Physical Table object. This overrides what you specified in the connection pool. ■ For an XML Server data source, you cannot specify an XSLT file on the XML tab of the Physical Table object.

To specify query output format settings

- 1 (Optional) For an XSLT file, type the path to and name of the XSLT file in the XSLT File field, or use the Browse button.
- 2 (Optional) For an XPath expression, type the XPath expression in the XPath Expression field, for example, `//XML`, or use the Browse button.

Setting Up the Persist Connection Pool Property

A persist connection pool is a database property that is used for specific types of queries (typically used to support Marketing queries). In some queries, all of the logical query cannot be sent to the transactional database because that database might not support all of the functions in the query. This issue might be solved by temporarily constructing a physical table in the database and rewriting the Analytics Server query to reference the new temporary physical table.

You could use the persist connection pool in the following situations:

- **Populate stored procedures.** Use to rewrite the logical SQL result set to a managed table. Typically used by the Siebel Marketing Server to write segmentation cache result sets.

- **Perform a generalized subquery.** Stores a nonfunction shipping subquery in a temporary table and then rewrites the original subquery result against this table. Reduces data movement between the Analytics Server and the database and supports unlimited IN list values and might result in improved performance.

NOTE: In these situations, the user issuing the logical SQL needs to have been granted the **Populate** privilege on the target database.

The persist connection pool functionality designates a connection pool with write-back capabilities for processing this type of query. You can assign one connection pool in a single database as a persist connection pool. If this functionality is enabled, the User name specified in the connection pool must have the privileges to create DDL (Data Definition Language) and DML (Data Manipulation Language) in the database.

Example of Using Buffer Size and Transaction Boundary

If each row size in a result set is 1 KB and the buffer size is 20 KB, then the maximum array size will be 20 KB. If there are 120 rows, there will be 6 batches with each batch size limited to 20 rows.

If you set the Transaction boundary field to 3, the server will commit twice. The first time the server commits after row 60 (3 * 20). The second time the server commits after row 120. If there is a failure when the server commits, the server will only rollback the current transaction. For example, if there are two commits and the first commit succeeds but the second commit fails, the server only rolls back the second commit. To make sure that the array-based insert runs successfully, it is recommended that you not set the transaction boundary greater than 10 and you set the buffer size to approximately 32 KB.

To assign a persist connection pool

- 1 In the Physical layer, double-click the database icon.
- 2 In the Database dialog box, click the General tab.
- 3 In the Persist Connection Pool area, click Set.

If there is only one connection pool, it appears in the Persist Connection Pool field.

- 4 If there are multiple connection pools, in the Browse dialog box, select the appropriate connection pool, and then click OK.

The selected connection pool name appears in the Persist connection pool field.

- 5 (Optional) To set write-back properties, click the Connection Pools tab.
- 6 In the connection pool list, double-click the connection pool.
- 7 In the Connection Pool dialog box, click the Write Back tab.
- 8 Complete the fields using [Table 12 on page 74](#) as a guide.
- 9 Click OK twice to save the persist connection pool.

To remove a persist connection pool

- 1 In the Physical layer, double-click the database icon.

2 In the Database dialog box, click the General tab.

3 In the Persist Connection Pool area, click Clear.

The database name is replaced by not assigned in the Persist connection pool field.

4 Click OK.

About Physical Tables

This topic is part of the [“Process of Creating the Physical Layer from Relational Data Sources” on page 58](#) and the [“Process of Creating the Physical Layer from Multidimensional Data Sources” on page 60](#).

A physical table is an object in the Physical layer of the Administration Tool that corresponds to a table in a physical database. Physical tables are usually imported from a database or another data source, and they provide the metadata necessary for the Analytics Server to access the tables with SQL requests.

In addition to importing physical tables, you can create virtual physical tables in the Physical layer, using the Object Type option in the Physical Table dialog box. A virtual physical table can be an alias, a stored procedure, or a Select statement. Virtual physical tables have several uses. You can use them to create the equivalent of a database view; that is, the virtual table generated by some query expression. You can use them to behave like a synonym; that is, a table identical to another table but with a different name. You can also use them to define a portion of the domain of a group of fragmented aggregate tables, as described in [“Define a Physical Layer Table with a Select Statement to Complete the Domain” on page 233](#). Creating virtual tables can provide the Analytics Server and the underlying databases with the proper metadata to perform some advanced query requests.

Object Types for Physical Tables

The Object Type drop-down list in the General tab of the Physical Table dialog box allows you to specify the physical table object type. [Table 15](#) provides a description of the available object types.

Table 15. Object Type Descriptions for Physical Tables

Object Type	Description
None	Specifies that the physical table object represents a physical table. If you select None, you can type a database hint. For more information, see “Using Database Hints” on page 99 .
Alias	Specifies that the physical table object is an alias to another table. When you select this option, the text pane to the right of the Object Type drop-down list becomes active, allowing you to type the alias name. The alias you type must be a valid physical table in the database specified in the connection pool. If you select Alias, you can type a database hint. For more information, see “Using Database Hints” on page 99 . If the database features table supports the COMMENT_START and COMMENT_END properties, the Analytics Server will include the alias name as a comment in the physical SQL it generates when it uses the alias as a data source.
Stored Proc	<p>Specifies that the physical table object is a stored procedure. When you select this option, the text pane to the right of the Object Type drop-down list becomes active, allowing you to type the stored procedure. Requests for this table will call the stored procedure.</p> <p>For stored procedures that are database specific, you can select the database type from the drop-down list above the text pane. At run time, if a stored procedure has been defined for the corresponding database's database type, then the stored procedure will be executed; otherwise, the default configuration will be executed.</p> <p>NOTE: Stored procedures using an Oracle database do not return result sets. For more information, see “Using Stored Procedures with an Oracle Database” on page 86.</p>
Select	<p>Specifies that the physical table object is a Select statement. When you select this option, the text pane to the right of the Object Type drop-down list becomes active, allowing you to type the select statement. Requests for this table will execute the Select statement.</p> <p>When you select this option, you need to manually create the table columns. The column names must match the ones specified in the Select statement. Column aliases are required for advanced SQL functions, such as aggregates and case statements.</p> <p>For Select statements that are database specific, you can select the database type from the drop-down list above the text pane. At run time, if a Select statement has been defined for the corresponding database's database type, then the Select statement will be executed; otherwise, the default configuration will be executed.</p>

Creating and Setting Up Physical Tables

This topic is part of the [“Process of Creating the Physical Layer from Relational Data Sources” on page 58](#) and the [“Process of Creating the Physical Layer from Multidimensional Data Sources” on page 60](#).

For all data sources, you can define general properties, columns, a primary key, and foreign keys.

Creating and Setting Up Physical Tables for Multidimensional Data Sources

Each cube from a multidimensional data source is set up as a physical cube table, a type of physical table. It has all the capabilities of a table such as columns (physical cube columns, not regular columns) and keys (optional) and foreign keys (optional). It also has cube-specific metadata such as hierarchies and levels.

In the Physical layer, a physical cube table looks like a regular table but will have a different icon. For more information about icons, see [“Icons and Symbols in the Administration Tool” on page 25](#).

When you import the physical schema, Siebel Business Analytics imports the cube, including its metrics, hierarchies, and levels. Although the hierarchies are imported, they are not displayed as such in the physical layer. Double-clicking on the cube icon brings up the various properties of the cube and provides access to the hierarchies.

Each multidimensional catalog can contain multiple physical cubes. You can import one or more of these cubes into your analytics repository. You can create a cube table manually. However, it is recommended that you import cube tables and their components.

NOTE: If creating a cube manually, it is strongly recommended that you build each cube one hierarchy at a time and test each one before building another. For example, create the time hierarchy and a measure, and then test it. When it is correct, create the geography hierarchy and test it. This will help make sure you have set up each cube correctly and make it easier to identify any setup errors.

Keys and foreign keys are not used for multidimensional data sources. When viewing the properties in the Physical Cube Table dialog box, the Hierarchies tab lists the various dimensional hierarchies in the cube. Users can drill down on any dimension to see the levels in that hierarchy.

To create a physical table or a physical cube table and any necessary properties, perform the following tasks:

- [Creating and Administering General Properties for a Physical Table on page 84](#)
- [Creating and Administering Columns and Keys in a Physical Table on page 86](#)
- [Setting Up Hierarchies in the Physical Layer for a Multidimensional Data Source on page 88](#)
- [Setting Physical Table Properties for an XML Data Source on page 92](#)

Creating and Administering General Properties for a Physical Table

Use the General tab of the Physical Table dialog box to create or edit a physical table in the Physical layer of the Administration Tool.

To create a physical table or edit general properties for the table

- 1 In the Physical layer of the Administration Tool, perform one of the following steps:
 - To create a physical table, right-click the physical database and choose New Object > Physical Table.
 - To create a physical cube table for a multidimensional data source, right-click the physical database and choose New Object > Physical Cube Table.
 - To edit an existing physical table, double-click the physical table icon.
- 2 In the selected Physical Table dialog box, complete the fields using [Table 16 on page 84](#) as a guide.

Table 16. Physical Table General Properties

Property	Description
Name	The administrator assigns a name to the new table.
Repository Object Name	<p>The name of the table and its associated path.</p> <ul style="list-style-type: none">■ When referencing an XML data source, this field displays the fully qualified name of a table used in an XML document.■ When referencing a multidimensional data source, this field displays the name of the source cube name from the multidimensional data source.
Make table cacheable	To include the table in the Analytics Server query cache, select the check box. When you select this check box, the Cache persistence time settings become active. This is useful for OLTP data sources and other data sources that are updated frequently. Typically, you should check this option for most tables.

Table 16. Physical Table General Properties

Property	Description
Cache persistence time	<p>How long table entries should persist in the query cache. The default value is Infinite, meaning that cache entries do not automatically expire. However, this does not mean that an entry will always remain in the cache. Other invalidation techniques, such as manual purging, LRU (Least Recently Used) replacement, metadata changes, and use of the cache polling table, result in entries being removed from the cache.</p> <p>If a query references multiple physical tables with different persistence times, the cache entry for the query will exist for the shortest persistence time set for any of the tables referenced in the query. This makes sure that no subsequent query gets a cache hit from an expired cache entry.</p> <p>If you change the default to minutes or seconds, type a whole number into the field on the left.</p> <p>For more information, see “Troubleshooting Problems with an Event Polling Table” on page 272.</p>
Object Type	<p>See Table 15 on page 82 for a description of the available object types. Depending on your selection, you may be presented with an additional drop-down list or a description field.</p>

Deleting a Physical Table

When you delete a physical table, all dependent objects are deleted. For example columns, keys, and foreign keys. When you delete a physical cube table, this also includes hierarchies.

To delete a physical table from the Physical layer

- 1 In the Physical layer of the Administration Tool, locate the table that you want to delete.
- 2 Right-click the table and choose Delete.

Viewing Data in Physical Tables or Columns

You can view the data in a physical table or an individual physical column by right-clicking on the object and choosing View Data. In online editing mode, you must check in changes before you can use View Data.

View Data is not available for physical cube tables or columns.

CAUTION: View Data will not work if you set the user name and password for connection pools to :USER and :PASSWORD. In offline mode, the Set values for variables dialog box appears so that you can populate :USER and :PASSWORD as part of the viewing process.

Using Stored Procedures with an Oracle Database

Stored Procedures within Oracle do not return result sets. Therefore they cannot be initiated from within Siebel Business Analytics. You need to rewrite the procedure as an Oracle function, use it in a select statement in the initialization block, and then associate it with the appropriate Siebel Business Analytics session variables in the Session Variables dialog box.

The function uses the GET_ROLES function and takes a user Id as a parameter and returns a semi-colon delimited list of group names.

The following is an example of an initialization SQL string using the GET_ROLES function that is associated with the USER, GROUP, DISPLAYNAME variables:

```
select user_id, get_roles(user_id), first_name || ' ' || last_name  
from csx_security_table  
where user_id = ':USER' and password = ':PASSWORD'
```

Creating and Administering Columns and Keys in a Physical Table

Each table in the Physical layer of the Administration Tool has one or more physical columns.

The Columns, Keys, and Foreign Keys tabs in the Physical Table dialog box allow you to view, create new, and edit existing columns, keys, and foreign keys that are associated with the table.

The following list describes the buttons that appear in the tabs:

- **New.** Opens the dialog box that corresponds to the tab.
- **Edit.** When you select an object and then click Edit, the dialog box that corresponds to the tab appears. You can then edit the object's properties.
- **Delete.** Deletes the selected object.

This section contains the following tasks:

- [Creating and Editing a Column in a Physical Table on page 86](#)
- [Specifying a Primary Key for a Physical Table on page 88](#)
- [Deleting a Physical Column For All Data Sources on page 88](#)

Creating and Editing a Column in a Physical Table

If the column is imported, the properties of the column are set automatically.

Measures in a Multidimensional Data Source

You need to select the aggregation rule for a physical cube column carefully to make sure your measures are correct. Setting it correctly might improve performance. Use the following guidelines to verify and assign the aggregation rule correctly:

- Verify aggregation rules after importing a cube. Typically, aggregation rules are assigned correctly when you import the cube. If a measure is a calculated measure in Analysis Services, the aggregation rule will be reported as None. Therefore, it is recommended that you examine the aggregation rule for all measures after importing a cube to verify that the aggregation rule has been assigned correctly.

For all measures assigned an aggregation rule value of None, contact the multidimensional data source administrator to verify that the value of the aggregation rule (Sum, Min, Max, and so on) is accurate. If you need to change the aggregation rule, you can change it in the Physical Cube Column dialog box.

- Setting the aggregation rule when you build the measures manually. Set the aggregation rule to match its definition in the multidimensional data source.

NOTE: Except when stated otherwise, the characteristics and behavior of a physical cube column are the same as for other physical columns.

To create or edit a physical column

- 1 In the Physical layer of the Administration Tool, perform one of the following steps:
 - To create a physical column, right-click a physical table and choose New Object > Physical Column from the shortcut menu.
 - To create a physical cube column for a multidimensional data source, right-click a physical cube table, and choose New Object > Physical Cube Column.
 - To edit an existing physical column, double-click the physical column icon.
- 2 In the Physical Column dialog box, type a name for the physical column.
For XML data sources, this field will store and display the unqualified name of a column (attribute) in an XML document.
- 3 In the Type field, select a data type for the physical column.
- 4 If applicable, specify the length of the data type.
For multidimensional data sources, if you select VARCHAR, you need to type a value in the Length field.
- 5 Select the Nullable option if the column is allowed to have null values.
The option Nullable in the Physical Columns dialog box indicates whether null values are allowed for the column. The data type list indicates the data type of the columns. This information is imported from the database when the column is imported to the Physical layer. It is generally safe to change a non-nullable value to a nullable value in a physical column. Making the value nullable allows null values to be returned to the user, which is expected with certain functions and with outer joins. Use caution in changing the data type values, however; setting the values to ones that are incorrect in the underlying data source might cause unexpected results.

If there are any data type mismatches, correct them in the repository or reimport the columns with mismatched data types. If you reimport columns, you also need to remap any logical column sources that reference the remapped columns. The data types of logical columns depend on the data types of physical columns that are their sources. The Analytics Server will furnish these logical column data types to client applications.

- 6 In the External Name field, type an external name.
 - Required if the same name (such as STATE) is used in multiple hierarchies.
 - Optional for XML documents and multidimensional data sources. The External Name field stores and displays the fully qualified name of a column (attribute).
- 7 (Multidimensional data sources) When the physical cube column is a measure, in the Aggregation rule drop-down list, select the appropriate value.

NOTE: A new physical cube column is created as a measure by default. To change this, see [“Setting Up Hierarchies in the Physical Layer for a Multidimensional Data Source” on page 88](#).

Specifying a Primary Key for a Physical Table

Use the Physical Key dialog box to specify the column or columns that define the primary key of the physical table.

To specify a primary key for a physical table

- 1 In the Physical layer of the Administration Tool, right-click a physical table and choose Properties.
- 2 In the Physical Table dialog box, click the Keys tab.
- 3 In the Keys tab, click New.
- 4 In the Physical Key dialog box, type a name for the key.
- 5 Select the check box for the column that defines the primary key of the physical table.
- 6 (Optional) In the Physical Key dialog box, type a description for the key, and then click OK.

Deleting a Physical Column For All Data Sources

You delete a physical column in the same way for all data sources.

NOTE: In a multidimensional data source, if you delete property or key columns from a level, the association is deleted and the column changes to a measure under the parent cube table.

To delete a physical column from the Physical layer

- 1 In the Physical layer of the Administration Tool, locate the column that you want to delete.
- 2 Right-click the column and choose Delete.

Setting Up Hierarchies in the Physical Layer for a Multidimensional Data Source

The following are some guidelines to follow when setting up hierarchies in the Physical layer.

- Hierarchies that are ragged or have a parent-child hierarchy are not imported. It is not possible to set up ragged or parent-child hierarchies in the physical layer.

- To change the column from a measure to a property or a level key, you need to set up a hierarchy and associate the cube column with the hierarchy. If you delete a property or level key column from a level, the column will change back to a measure under the parent cube table.

CAUTION: You will need to build a matching hierarchy in the Business Model and Mapping layer. If you do not do this, queries may appear to work but might not return the correct results.

- Set up hierarchies for large cubes (those with many members on the lowest levels of the hierarchy) to minimize performance issues. It is recommended that you do not set up hierarchies and levels unless they are required to run queries.

To create and maintain hierarchies in the Physical Layer, perform the following tasks:

- [Adding a Hierarchy to a Physical Cube Table on page 89](#)
- [Verifying Hierarchy Levels on page 90](#)
- [Adding or Removing a Cube Column in an Existing Hierarchy on page 91](#)
- [Removing a Hierarchy from a Physical Cube Table on page 91](#)
- [Associating a Physical Cube Column with a Hierarchy Level on page 92](#)

Adding a Hierarchy to a Physical Cube Table

Each level in a hierarchy has a level key. The first cube column associated with (added to) the level of a hierarchy is the level key. This must match with the data source definition of the cube. The data source cube table cannot set one column as a level key and the Siebel Business Analytics physical layer table set a different column as a level key. The icon for the column that you select first changes to the key icon after it is associated with the level of a hierarchy.

When you select columns to include in a hierarchy, it is recommended that you select them in hierarchical order, starting with the highest level. After adding columns to the hierarchy, you can change the sequence.

If you select multiple columns and bring them into the hierarchy at the same time, the order of the selected group of columns remains the same. You can change the order of the columns in the Browse dialog box.

If a member of a hierarchy is not explicitly referred to in a query, a default member must be used. Therefore, every hierarchy is associated with a default member, typically the ALL member. The Hierarchy dialog box contains a check box (Default member type ALL) that you use when you want to designate the ALL member as the default. The following list contains some guidelines about selecting the check box:

- If you import the cube, the Default member type ALL check box should be automatically selected. The ALL member is identified during import.
- If you build the hierarchies manually, the check box will not be automatically selected. Before selecting the check box, ask your multidimensional data source administrator if a non-ALL default member has been defined. For example, for the Year level, 1997 might be designated as the default member. In this case, the Default member type ALL check box would not be checked.

To add a hierarchy to a physical cube table

- 1 In the Physical layer of the Administration Tool, double-click the table to which you want to add a hierarchy.
- 2 In the Physical Cube Table dialog box, click the Hierarchies tab and click Add.
- 3 In the Hierarchy dialog box, in the Name field, type a name for the hierarchy, and click Add.
NOTE: In a hierarchy, levels should be added from the top down (you can reorder them later). Using the correct hierarchical sequence allows your queries to return accurate information and avoids errors.
- 4 To create a level, in the Physical Level dialog box, in the Name field, type a name for the level, and click Add.
NOTE: You can also add columns to a physical level by dragging and dropping physical columns on the level object. The first column you add will be a key. Subsequent columns will be properties.
- 5 To add columns to the level, in the Browse dialog box, perform the following steps:
 - a In the Name list, locate the columns that you want to add to the hierarchy.
 - b Add the key column first.
 - c Add other columns by pressing Ctrl on your keyboard while clicking each column, and then clicking Select.
- 6 In the Physical Level dialog box, click OK.
- 7 In the Hierarchy dialog box, in the Levels tab, the Default member type ALL check box should not be selected for non-ALL default members.

Verifying Hierarchy Levels

It is strongly recommended that after setting up a hierarchy containing more than one level, you should verify the order of the levels in a hierarchy.

To verify the levels in a hierarchy

- 1 In the Physical layer of the Administration Tool, double-click the table you want to verify.
- 2 In the Physical Cube Table dialog box, click the Hierarchies tab.
- 3 In the Hierarchies tab, select a hierarchy, and then click Edit.
- 4 In the Hierarchy dialog box, verify the levels are correct.
The Hierarchy dialog box lists all the defined levels for the selected hierarchy. The highest level in the hierarchy should be the first (highest) item in the list.
- 5 If you need to reorder the hierarchy levels, select a level and click Up or Down to correct the order of the levels.
There must be multiple levels and you must select a level for the buttons to be available.
- 6 When the levels are correct, click OK.
- 7 In the Physical Cube Table dialog box, click OK.

Adding or Removing a Cube Column in an Existing Hierarchy

After setting up a hierarchy you may need to add or remove a column. You might want to remove a hierarchy if it has been built incorrectly and you want to start over.

If you remove a cube column from a hierarchy, it is deleted from the hierarchy but remains in the cube table and is available for selection to add to other levels.

To add a cube column to or remove a cube column from an existing hierarchy

- 1 In the Physical layer of the Administration Tool, double-click the table that you want to change.
- 2 In the Physical Cube Table dialog box, click the Hierarchies tab.
- 3 Select the hierarchy you want to change, and then click Edit.
- 4 In the Hierarchy dialog box, select the level and click Edit.
- 5 In the Physical Level dialog box, perform one of the following steps:
 - a To add a column, click Add.
 - ❑ In the Browse dialog box, in the Name list, click to select the columns that you want to add.
 - ❑ Click Select and then click OK three times to return to the Physical layer in the Administration Tool.
 - b To remove a column, select the column and click Remove.
 - c To change the sequence of the levels in a hierarchies, select the level and click Up or Down.
 - d Click OK.
- 6 In the Hierarchy dialog box, click OK.
- 7 In the Physical Cube Table dialog box, click OK.

Removing a Hierarchy from a Physical Cube Table

You might want to remove a hierarchy if it has been built incorrectly and you want to start over or to remove objects that are not accessed. Perhaps the business model does not reference any of the elements in that hierarchy. For example, you import an entire physical multidimensional schema and only want to keep parts of it in the business model.

NOTE: When you delete a hierarchy in the Physical layer, you remove the hierarchy and the columns that are part of the hierarchy. This is different from deleting a hierarchy in the Business Model and Mapping layer.

To remove a hierarchy from a physical cube table

- 1 In the Physical layer of the Administration Tool, double-click the table that you want to change.
- 2 In the Physical Cube Table dialog box, click the Hierarchies tab.
- 3 Select the hierarchy you want to remove, and then click Remove.

Associating a Physical Cube Column with a Hierarchy Level

Attributes are used in the physical layer to represent columns that only exist at a particular level of a hierarchy. For example, if Population is an attribute that is associated with the level State in the Geography hierarchy, when you query for Population you are implicitly asking for data that is at the State level in the hierarchy.

There can be zero or more attributes associated with a level. The first physical cube column that is associated with a level becomes the level key. If you associate subsequent columns with a level, they become attributes, not level keys.

Example of Associating a Physical Cube Column with a Hierarchy

You have a level called State and you want to associate a column called Population with this level.

- Create the hierarchy and the State level.
- Create the physical cube column for Population.
- In the Physical Cube Table dialog box, in the Hierarchies tab, select the State level and click Edit.
- In the Hierarchy dialog box, click Add.
- In the Physical Level dialog box, click Add.
- In the Browse dialog box, select the Population column and click Select.

The measure icon changes to the property icon.

Setting Physical Table Properties for an XML Data Source

Use the XML tab to set or edit properties for an XML data source. The XML tab of the Physical Table dialog box provides the same functionality as the XML tab of the Connection Pool dialog box. However, setting properties in the Physical Table dialog box will override the corresponding settings in the Connection Pool dialog box. For more information, see [“Setting Up Additional Connection Pool Properties for an XML Data Source” on page 77](#).

Creating Physical Layer Folders

This section contains the following topics:

- [Creating Catalog Folders on page 93](#)
- [Creating Schema Folders on page 93](#)
- [Using a Variable to Specify the Name of a Catalog or Schema on page 93](#)
- [Setting Up Display Folders in the Physical Layer on page 94](#)

Creating Catalog Folders

A catalog folder contains one or more physical schema folders. Catalog folders are optional in the Physical layer of the Administration Tool.

Creating a Catalog

Use the General tab of the Catalog dialog box to define or edit a catalog object in the Physical layer of the Administration Tool. You must create a database object before you create a catalog object.

To create a catalog

- 1 In the Physical layer, right-click a database, and then choose New Object > Physical Catalog.
- 2 In the Physical Catalog dialog box, type a name for the catalog.
- 3 Type a description for the catalog, and then click OK.

Creating Schema Folders

The schema folder contains tables and columns for a physical schema. Schema objects are optional in the Physical layer of the Administration Tool.

You must create a database object before you create a schema object.

To create a schema folder

- 1 In the Physical layer, right-click a database, and then choose New Object > Physical Schema.
- 2 In the Physical Schema dialog box, type a name.
- 3 Type a description for the schema, and then click OK.

Using a Variable to Specify the Name of a Catalog or Schema

You can use a variable to specify the names of catalog and schema objects. For example, you have data for multiple clients and you structured the database so that data for each client was in a separate catalog. You would initialize a session variable named Client, for example, that could be used to set the name for the catalog object dynamically when a user signs on to the Analytics Server.

NOTE: The Dynamic Name tab is not active unless at least one session variable is defined.

To specify the session variable to use in the Dynamic Name tab

- 1 In the Name column of the Dynamic Name tab, click the name of the session variable that you want to use. The initial value for the variable (if any) is shown in the Default Initializer column.

- 2 To select the highlighted variable, click Select.

The name of the variable appears in the dynamic name field, and the Select button toggles to the Clear button.

To remove assignment for a session variable in the Dynamic Name tab

- Click Clear to remove the assignment for the variable as the dynamic name.

The value Not Assigned displays in the dynamic name field, and the Clear button toggles to the Select button.

To sort column entries in the Dynamic Name tab

- You can sort the entries in a column by clicking on the associated column heading, Name or Default Initializer. Clicking on a column heading toggles the order of the entries in that column between ascending and descending order, according to the column type.

When no dynamic name is assigned, Not Assigned displays in the dynamic name field to the left of the Select button. When a dynamic name is assigned, the Select button toggles to the Clear button, and the name of the variable displays in the dynamic name field.

Setting Up Display Folders in the Physical Layer

Administrators can create display folders to organize table objects in the Physical layer. They have no metadata meaning. After you create a display folder, the selected tables appear in the folder as a shortcut and in the Physical layer tree as an object. You can hide the objects so that you only see the shortcuts in the display folder. For more information about hiding these objects, see [“Using the Options Dialog Box—Repository Tab” on page 34](#).

NOTE: Deleting objects in the display folder deletes only the shortcuts to those objects.

To set up a physical display folder

- 1 In the Physical layer, right-click a database object, and choose New Object > Physical Display Folder.
- 2 In the Physical Display Folder dialog box, in the Tables tab, type a name for the folder.
- 3 To add tables to the display folder, perform the following steps:
 - a Click Add.
 - b In the Browse dialog box, select the fact or physical tables you want to add to the folder and click Select.
- 4 Click OK.

About Physical Joins

All valid physical joins need to be configured in the Physical layer of the Administration Tool.

NOTE: You do not create joins for multidimensional data sources.

When you import keys in a physical schema, the primary key-foreign key joins are automatically defined. Any other joins within each database or between databases have to be explicitly defined. You need to define any joins that express relationships between tables in the physical layer of the repository.

NOTE: Imported key and foreign key joins do not have to be used in metadata. Joins that are defined to enforce referential integrity constraints can result in incorrect joins being specified in queries. For example, joins between a multipurpose lookup table and several other tables can result in unnecessary or invalid circular joins in the SQL issued by the Analytics Server.

Multi-Database Joins

A multi-database join is defined as a table under one metadata database object that joins to a table under a different metadata database object. You need to specify multi-database joins to combine the data from different databases. Edit the Physical Table Diagram window to specify multi-database joins. The joins can be between tables in any databases, regardless of the database type, and are performed within the Analytics Server. While the Analytics Server has several strategies for optimizing the performance of multi-database joins, multi-database joins will be significantly slower than joins between tables within the same database. It is recommended to avoid them whenever possible. For more information about the Physical Table Diagram, see [“Defining Physical Joins in the Physical Diagram” on page 97](#).

Fragmented Data

Fragmented data is data from a single domain that is split between multiple tables. For example, a database might store sales data for customers with last names beginning with the letter A through M in one table and last names from N through Z in another table. With fragmented tables, you need to define all of the join conditions between *each* fragment and all the tables it relates to. [Figure 9](#) shows the physical joins with a fragmented sales table and a fragmented customer table where they are fragmented the same way (A through M and N through Z).

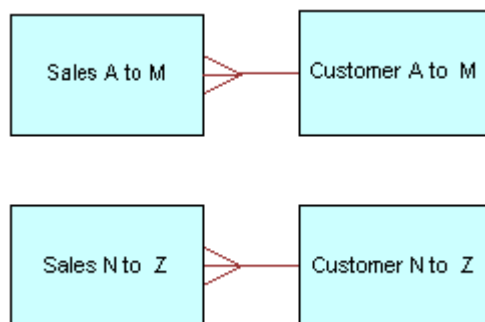


Figure 9. Fragmented Tables Example

In some cases, you might have a fragmented fact table and a fragmented dimension table, but the fragments might be across different values. In this case, you need to define all of the valid joins, as shown in [Figure 10](#).

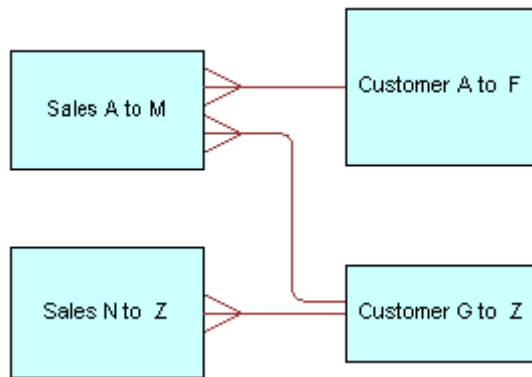


Figure 10. Joins for Fragmented Tables Example

TIP: Avoid adding join conditions where they are not necessary (for example, between Sales A to M and Customer N to Z in [Figure 9](#)). Extra join conditions can cause performance degradations.

Primary Key and Foreign Key Relationships

A primary key and foreign key relationship defines a one-to-many relationship between two tables. A foreign key is a column or a set of columns in one table that references the primary key columns in another table. The primary key is defined as a column or set of columns where each value is unique and identifies a single row of the table. You can specify primary key and foreign keys in the Physical Table Diagram or by using the Keys tab and Foreign Keys tab of the Physical Table dialog box. See also “[Defining Physical Joins in the Physical Diagram](#)” on page 97 and “[Creating and Administering Columns and Keys in a Physical Table](#)” on page 86.

Complex Joins

Complex joins are joins over nonforeign key and primary key columns. When you create a complex join in the Physical layer, you can specify expressions and the specific columns on which to create the join. When you create a complex join in the Business Model and Mapping layer, you do not specify expressions.

Defining Physical Foreign Keys and Joins

You can create physical complex joins and logical joins using the Joins Manager or using the Physical or Logical Table Diagram.

NOTE: You do not create joins for multidimensional data sources.

To define physical joins, see the following topics:

- [Defining Physical Foreign Keys or Complex Joins with the Joins Manager on page 97](#)
- [Defining Physical Joins in the Physical Diagram on page 97](#)

Defining Physical Foreign Keys or Complex Joins with the Joins Manager

You can use the Joins Manager to view join relationships and to create physical foreign keys and complex joins.

To create a physical foreign key or complex join

- 1 In the Administration Tool toolbar, select **Manage > Joins**.
- 2 In the Joins Manager dialog box, perform one of the following tasks:
 - Select **Action > New > Complex Join**.
The Physical Complex Join dialog box appears.
 - Select **Action > New > Foreign Key**. In the Browse dialog box, double-click a table.
- 3 In the Physical Foreign Key dialog box, type a name for the foreign key.
- 4 Click the Browse button for the Table field on the left side of the dialog box, and then locate the table that the foreign key references.
- 5 Select the columns in the left table that the key references.
- 6 Select the columns in the right table that make up the foreign key columns.
- 7 If appropriate, specify a database hint.
For more information, see [“Using Database Hints” on page 99](#).
- 8 To open the Expression Builder, click the button to the right of the Expression pane.
The expression displays in the Expression pane.
- 9 Click OK to save your work.

Defining Physical Joins in the Physical Diagram

The Physical Diagram window shows physical tables and any defined joins. You can use the Physical Table Diagram window to define foreign keys and complex joins between tables, whether or not the tables are in the same database.

If you click the Physical diagram icon on the toolbar, the Physical Table Diagram window opens and only the selected objects appear. If you right-click a physical object, several options are available. For more information about these options, see [Table 17](#).

To display the Physical Table Diagram

- 1 In the Administration Tool, in the Physical layer, right-click a table and choose Physical Diagram.
- 2 In the shortcut menu, choose an option from [Table 17](#).
- 3 To add another table to the Physical Table Diagram window, perform the following steps:
 - a Leave the Physical Table Diagram window open.
 - b Right-click to select a table you want to add and choose one of the Physical Diagram options described in [Table 17](#).

Repeat this process until all the tables you need appear in the Physical Table Diagram window.

Table 17. Physical Diagram Shortcut Menu Options

Physical Diagram Menu	Description
Selected object(s) only	Displays the selected objects only. Joins display only if they exist between the objects that you select.
Object(s) and direct joins	Displays the selected objects and any tables that join to the objects that you select.
Object(s) and all joins	Displays the selected objects, as well as each object that is related directly or indirectly to the selected object through some join path. If all the objects in a schema are related, then using this option diagrams every table, even if you only select one table.

To define a foreign key join or a complex join

- 1 In the Physical layer of the Administration Tool, select one or more tables and execute one of the Physical Diagram commands from the right-click menu.
- 2 Click one of the following icons on the Administration Tool toolbar:
 - New foreign key
 - New complex join
- 3 With this icon selected, in the Physical Table Diagram window, left-click the first table in the join (the table representing **one** in the one-to-many join) to select it.
- 4 Move the cursor to the table to which you want to join (the table representing **many** in the one-to-many join), and then left-click the second table to select it.

The Physical Foreign Key or Physical Join dialog box appears.
- 5 Select the joining columns from the left and the right tables.

The SQL join conditions appear in the expression pane of the window.

NOTE: The driving table feature is shown on this window, but it is not available for selection because the Analytics Server implements driving tables only in the Business Model and Mapping layer. For more information about driving tables, see [“Specifying a Driving Table”](#) on page 132.

- 6 If appropriate, specify a database hint.

For more information, see [“Using Database Hints” on page 99](#).

- 7 To open the Expression Builder, click the button to the right of the Expression pane.

The expression displays in the Expression pane.

- 8 Click OK to apply the selections.

Using Database Hints

Database hints are instructions placed within a SQL statement that tell the database query optimizer the most efficient way to execute the statement. Hints override the optimizer's execution plan, so you can use hints to improve performance by forcing the optimizer to use a more efficient plan.

NOTE: Hints are database specific. Analytics Server supports hints only for Oracle 8i and 9i servers.

Using the Administration Tool, you can add hints to a repository, in both online and offline modes, to optimize the performance of queries. When you add a hint to the repository, you associate it with database objects. When the object associated with the hint is queried, the Analytics Server inserts the hint into the SQL statement.

Table 18 shows the database objects with which you can associate hints. It also shows the Administration Tool dialog box that corresponds to the database object. Each of these dialog boxes contains a Hint field, into which you can type a hint to add it to the repository.

Table 18. Database Objects That Accept Hints

Database Object	Dialog Box
Physical table - object type of None	Physical Table - General tab
Physical table - object type of Alias	Physical Table - General tab
Physical foreign key	Physical Foreign Key
Physical complex join	Physical Join - Complex Join

Usage Examples

This section provides a few examples of how to use Oracle hints in conjunction with the Analytics Server. For more information about Oracle hints, see the following Oracle documentation: *Oracle8i Tuning* for reference information, and *Oracle8i SQL Reference* for descriptions about Oracle hints and hint syntax.

Index Hint

The Index hint instructs the optimizer to scan a specified index rather than a table. The following hypothetical example explains how you would use the Index hint. You find queries against the ORDER_ITEMS table to be slow. You review the query optimizer's execution plan and find the FAST_INDEX index is not being used. You create an Index hint to force the optimizer to scan the FAST_INDEX index rather than the ORDER_ITEMS table. The syntax for the Index hint is *index(table_name, index_name)*. To add this hint to the repository, navigate to the Administration Tool's Physical Table dialog box and type the following text in the Hint field:

```
i ndex(ORDER_I TEMS, FAST_I NDEX)
```

Leading Hint

The Leading hint forces the optimizer to build the join order of a query with a specified table. The syntax for the Leading hint is *leading(table_name)*. If you were creating a foreign key join between the Products table and the Sales Fact table and wanted to force the optimizer to begin the join with the Products table, you would navigate to the Administration Tool's Physical Foreign Key dialog box and type the following text in the Hint field:

```
L eadi ng(Products)
```

Performance Considerations

Hints that are well researched and planned can result in significantly better query performance. However, hints can also negatively affect performance if they result in a suboptimal execution plan. The following guidelines are provided to help you create hints to optimize query performance:

- You should only add hints to a repository after you have tried to improve performance in the following ways:
 - Added physical indexes (or other physical changes) to the Oracle database.
 - Made modeling changes within the server.
- Avoid creating hints for physical table and join objects that are queried often.

NOTE: If you drop or rename a physical object that is associated with a hint, you must also alter the hints accordingly.

Creating Hints

The following procedure provides the steps to add hints to the repository using the Administration Tool.

To create a hint

- 1 Navigate to one of the following dialog boxes:
 - Physical Table—General tab
 - Physical Foreign Key
 - Physical Join—Complex Join

- 2 Type the text of the hint in the Hint field and click OK.

For a description of available Oracle hints and hint syntax, see *Oracle8i SQL Reference*.

NOTE: Do not type SQL comment markers (/* or --) when you type the text of the hint. The Analytics Server inserts the comment markers when the hint is executed.

5

Creating and Administering the Business Model and Mapping Layer in a Repository

The following topics contain information about creating the business model and logical objects:

- [Creating Business Model Objects on page 104](#)
- [Duplicate Business Model and Presentation Catalog on page 104](#)
- [Creating and Administering Logical Tables on page 105](#)
- [Creating and Administering Logical Columns on page 107](#)
- [Creating and Administering Logical Table Sources \(Mappings\) on page 111](#)
- [About Dimensions and Hierarchical Levels on page 119](#)
- [Process of Creating and Administering Dimensions on page 119](#)
- [Setting Up Display Folders in the Business Model and Mapping Layer on page 128](#)
- [Defining Logical Joins on page 129](#)

About Creating the Business Model and Mapping Layer

This section is part of the roadmap for planning and setting up a repository. For more information, see [“Planning and Creating a Siebel Business Analytics Repository” on page 41](#).

After creating all of the elements of the Physical layer, you can drag tables or columns in the Physical layer to the Business Model and Mapping layer. For more information, see [“Creating and Administering the Physical Layer in a Repository” on page 57](#) and [“Creating the Business Model Layer for a Multidimensional Data Source” on page 104](#).

The Business Model and Mapping layer of the Administration Tool defines the business, or logical, model of the data and specifies the mapping between the business model and the physical layer schemas.

You create one or more business models in the logical layer and create logical tables and columns in each business model. To automatically map objects in the Business Model and Mapping layer to sources in the Physical layer, you can drag and drop Physical layer objects to a business model in the logical layer. When you drag a physical table to the Business Model and Mapping layer, a corresponding logical table is created. For each physical column in the table, a corresponding logical column is created. If you drag multiple tables at once, a logical join is created for each physical join, but only the first time the tables are dragged onto a new business model.

Creating the Business Model Layer for a Multidimensional Data Source

Setting up the Business Model and Mapping (logical) layer for multidimensional data sources is similar to setting up the logical layer for a relational data source. To create the business model layer, you can drag and drop the physical layer cube to the logical layer. However, because the contents of the physical cube are added as one logical table, you still have to reorganize the columns into appropriate logical tables and recreate the hierarchies.

Creating Business Model Objects

The Business Model and Mapping layer of the Administration Tool can contain one or more business model objects. A business model object contains the business model definitions and the mappings from logical to physical tables for the business model.

NOTE: When you work in a repository in offline mode, remember to save your repository from time to time. You can save a repository in offline mode even though the business models may be inconsistent.

To create a business model

- 1 Right-click in the Business Model and Mapping layer below the existing objects.
- 2 Select the option New Business Model from the shortcut menu.
- 3 Specify a name for the business model or accept the default.
- 4 If the business model is available for queries, select the option Available for queries.

NOTE: The business model should be consistent before you make it available for queries.

- 5 (Optional) Type a description of the business model, and then click OK.

Duplicate Business Model and Presentation Catalog

This allows you to select a matching business model and presentation catalog, make a copy, and assign new names to the duplicates.

NOTE: Aliases are not copied.

To copy a business model and its presentation catalog

- 1 In the Business Model and Mapping layer, right-click a business model.
- 2 In the right-click menu, choose Duplicate with Presentation Catalog.
- 3 In the Copy Business Model and Presentation Catalog dialog box, select the business model to copy.

- 4 Specify new names for the business model and its catalog in the appropriate name fields, and then click OK.

The copied business model appears in the Business Model and Mapping layer window.

Creating and Administering Logical Tables

Logical tables exist in the Business Model and Mapping layer. The logical schema defined in each business model needs to contain at least two logical tables and you need to define relationships between them.

Each logical table has one or more logical columns and one or more logical table sources associated with it. You can change the logical table name, reorder the sources, and configure the logical keys (primary and foreign).

This section includes the following topics:

- [Creating Logical Tables on page 105](#)
- [Specifying a Primary Key in a Logical Table on page 106](#)
- [Reviewing Foreign Keys for a Logical Table on page 107](#)

Creating Logical Tables

Typically, you create logical tables by dragging and dropping a physical table from the Physical layer to a business model in the Business Model and Mapping layer. If a table does not exist in your physical schema, you would need to create the logical table manually.

Drag and drop operations are usually the fastest method for creating objects in the Business Model and Mapping layer. If you drag and drop physical tables from the Physical layer to the Business Model and Mapping layer, the columns belonging to the table are also copied. After you drag and drop objects into the Business Model and Mapping layer, you can modify them in any way necessary without affecting the objects in the Physical layer.

When you drag physical tables (with key and foreign key relationships defined) to a business model, logical keys and joins are created that mirror the keys and joins in the physical layer. This occurs only if the tables that you drag include the table with the foreign keys. Additionally, if you create new tables or subsequently drag additional tables from the Physical layer to the Business Model and Mapping layer, the logical links between the new or newly dragged tables and the previously dragged tables must be created manually.

For more information about joins, see [“Defining Logical Joins with the Joins Manager” on page 130](#) and [“Defining Logical Joins with the Business Model Diagram” on page 131](#).

To create a logical table by dragging and dropping

- 1 Select one or more table objects in the Physical layer.

You must include the table with the foreign keys if you want to preserve the keys and joins from the physical layer.

- 2 Drag and drop the table objects to a business model in the Business Model and Mapping layer.

When you drop them, the table objects, including the physical source mappings, are created automatically in the Business Model and Mapping layer.

To create a logical table manually

- 1 In the Business Model and Mapping layer, right-click the business model in which you want to create the table and select New Object > Logical Table.

The Logical Table dialog box appears.

- 2 In the General tab, type a name for the logical table.

- 3 If this is a bridge table, select the option Bridge table.

For more information, see [“Identifying Dimension Hierarchies” on page 45](#).

- 4 (Optional) Type a description of the table.

- 5 Click OK.

NOTE: After creating a logical table manually, you must create all keys and joins manually.

Adding or Editing Logical Table Sources

You can add a new logical table source, edit or delete an existing table source, create or change mappings to the table source, and define when to use logical tables sources and how content is aggregated. For instructions about how to perform these tasks, see [“Creating and Administering Logical Table Sources \(Mappings\)” on page 111](#).

Specifying a Primary Key in a Logical Table

After creating tables in the Business Model and Mapping layer, you specify a primary key for each table. Logical dimension tables must have a logical primary key. Logical keys can be composed of one or more than one logical columns.

NOTE: Logical keys are optional for logical fact tables. However, it is recommended that you do not specify logical keys for logical fact tables. For more information, see [“Reviewing Foreign Keys for a Logical Table” on page 107](#).

To specify a primary key in a logical table

- 1 In the Business Model and Mapping layer, double-click a table.
- 2 In the Logical Table dialog box, select the Keys tab and then click New.

- 3 In the Logical Key dialog box, perform the following steps:
 - a Type a name for the key.
 - b Select the check box for the column that defines the key of the logical table.
- 4 Click OK.

Reviewing Foreign Keys for a Logical Table

You can use the Foreign Keys tab to review the foreign keys for a logical table.

CAUTION: It is recommended that you do not have foreign keys for logical tables. However, you can create logical foreign keys and logical complex joins using either the Joins Manager or the Business Model Diagram. A logical key for a fact table must be made up of the key columns that join to the attribute tables. For more information, see [“Defining Logical Joins” on page 129](#).

Joins and Fact Tables

In fact tables, it is recommended that you use complex logical joins instead of foreign key logical joins. If complex logical joins are used, then there is more flexibility in defining the primary key. If the physical table has a primary key, then this field can be used as a logical key for the fact table.

NOTE: This is the method recommended for the analytics repository.

To review foreign key information for a logical table

- 1 In the Business Model and Mapping layer, double-click a table.
- 2 In the Logical Table dialog box, select the Foreign Keys tab.
- 3 To review an existing foreign key, in the Foreign Keys list, select a key and click Edit.

The Logical Foreign Key dialog box appears. For more information about changing information in this dialog box, see [“Defining Logical Joins” on page 129](#).

Creating and Administering Logical Columns

Many logical columns are automatically created by dragging tables from the Physical layer to the Business Model and Mapping layer. Other logical columns, especially ones that involve calculations based on other logical columns, can be created later.

Logical columns are displayed in a tree structure expanded out from the logical table to which they belong. If the column is a primary key column or participates in a primary key, the column is displayed with the key icon. If the column has an aggregation rule, it is displayed with a sigma icon. You can reorder logical columns in the Business Model and Mapping layer.

This section includes the following topics:

- [Creating and Moving a Logical Column](#)

- [Setting Default Levels of Aggregation for Measure Columns on page 109](#)
- [Associating an Attribute with a Logical Level in Dimension Tables on page 110](#)

Creating and Moving a Logical Column

Use the General tab to create or edit the general properties of a logical column. You can create a logical column object in the Business Model and Mapping layer, and then drag and drop it to the Presentation layer.

About Sorting on a Logical Column

For a logical column, you can specify a different column on which to base the sort. This changes the sort order of a column when you do not want to order the values *lexicographically*. Lexicographical sort arranges the results in alphabetic order such as in a dictionary. In this type of sort, numbers are ordered by their alphabetic spelling and not divided into a separate group.

For example, if you sorted on month (using a column such as MONTH_NAME), the results would be returned as February, January, March, and so on, in lexicographical sort order. However, you might want months to be sorted in chronological order. Therefore, your table should have a month key (such as MONTH_KEY) with values of 1 (January), 2 (February), 3 (March), and so on. To achieve the desired sort, you set the Sort order column field for the MONTH_NAME column to be MONTH_KEY. Then a request to order by MONTH_NAME would return January, February, March, and so on.

To create a logical column

- 1 In the Business Model and Mapping layer, right-click a logical table.
- 2 From the shortcut menu, select New Object > Logical Column.
- 3 In the Logical Column dialog box, select the General tab.
- 4 In the General tab, type a name for the logical column.

The name of the business model and the associated logical table appear in the Belongs to Table field.

- 5 (Optional) If you want to assign a different column on which to base the sort order for a column, perform the following steps:
 - a Next to the Sort order column field, click Set.
 - b In the Browse dialog box, select a column
 - c To view the column details, click View to open the Logical Column dialog box for that column, and then click Cancel.
NOTE: You can make some changes in this dialog box. If you make changes, click OK to accept the changes instead of Cancel.
 - d In the Browse dialog box, Click OK.
- 6 (Optional) To remove the Sort order column value, click Clear.

- 7 (Optional) If you want the logical column to be derived from other logical columns, perform the following steps:
 - a Select the check box for Use existing logical columns as source.
 - b Click the ellipsis button next to the text box to open the Expression Builder.
 - c In the Expression Builder- derived logical column dialog box, specify the expression from which the logical column should be derived.
 - d Click OK.
- 8 (Optional) In the Logical Column dialog box, type a description of the logical column.

NOTE: The type and length fields are populated automatically based upon the column's source.
- 9 Click OK.

To move logical columns

- 1 In the Business Model and Mapping layer, drag and drop a logical column to a different logical table.

NOTE: You can select multiple columns to move.
- 2 In the Sources for moved columns dialog box, in the Action area, select an action.
- 3 If you select Ignore, no logical source will be added in the Sources folder of the destination table.
- 4 If you select Create new, a copy of the logical source associated with the logical column will be created in the Sources folder of the destination table.
- 5 If you select Use existing, in the Use existing drop-down list, you must select a logical source from the Sources folder of the destination table.

The column that you moved will now be associated with this logical source.

Setting Default Levels of Aggregation for Measure Columns

You need to specify aggregation rules for mapped logical columns that are measures. Aggregation should only be performed on measure columns. Measure columns should exist only in logical fact tables.

You can specify an override aggregation expression for specific logical table sources. This helps the Analytics Server take advantage of aggregate tables when the default aggregation rule is Count Distinct. If you do not specify any override, then the default rule prevails.

To specify a default aggregation rule for a measure column

- 1 In the Business Model and Mapping layer, double-click a logical column.
- 2 In the Logical Column dialog box, click the Aggregation tab.

- 3 In the Aggregation tab, select one of the aggregate functions from the Default Aggregation Rule drop-down list.

The function you select is always applied when an end user or an application requests the column in a query.

- 4 Click OK.

Associating an Attribute with a Logical Level in Dimension Tables

Attributes can be associated with a logical level by selecting the dimensional level on the Levels tab. Measures can be associated with levels from multiple dimensions and will always aggregate to the levels specified.

Dimensions appear in the Dimensions list. If this attribute is associated with a logical level, the level appears in the Levels list.

Another way to associate a measure with a level in a dimension is to expand the dimension tree in the Business Model and Mapping layer, and then use the drag-and-drop feature to drop the column on the target level. For more information about level-based measures, see [“Level-Based Measure Calculations Example” on page 123](#).

To associate this measure with a logical level in a dimension

- 1 In the Business Model and Mapping layer, double-click a logical column.
- 2 In the Logical Column dialog box, click the Levels tab.
- 3 In the Levels tab, click the Logical Levels field for the dimension from which you want to select a logical level.

NOTE: In the Levels tab, in the levels list, you can sort the rows (toggle between ascending order and descending order) by clicking a column heading.

- 4 In the Logical Levels drop-down list, select the level.
- 5 Repeat this process to associate this measure with other logical levels in other dimensions.

To remove the association between a dimension and a measure

- 1 In the Business Model and Mapping layer, double-click a logical column.
- 2 In the Logical Column dialog box, click the Levels tab.
- 3 In the Levels tab, click the delete button next to the Logical Levels field.
- 4 Click OK.

Creating and Administering Logical Table Sources (Mappings)

You can add a new logical table source, edit or delete an existing table source, create or change mappings to the table source, and define when to use logical tables sources and how content is aggregated. Additionally, you can copy aggregation content to the Windows clipboard or from another logical table source, and check the aggregation content of logical fact table sources.

You would add new logical table sources when more than one physical table could be the source of information. For example, many tables could hold information for revenue. You might have three different business units (each with its own order system) where you get revenue information. In another example, you might periodically summarize revenue from an orders system or a financial system and use this table for high-level reporting.

One logical table source folder exists for each logical table. The folder contains one or more logical table sources. These sources define the mappings from the logical table to the physical table. Complex mappings, including formulas, are also configured in the logical table sources.

Logical tables can have many physical table sources. A single logical column might map to many physical columns from multiple physical tables, including aggregate tables that map to the column if a query asks for the appropriate level of aggregation on that column.

When you create logical tables and columns by dragging and dropping from the Physical layer, the logical table sources are generated automatically. If you create the logical tables manually, you need to also create the sources manually.

For examples of how to set up fragmentation content for aggregate navigation, see [“Specify Fragmentation Content” on page 227](#).

This section includes the following topics:

- [Creating or Removing a Logical Table Source on page 111](#)
- [Defining Physical to Logical Table Source Mappings on page 113](#)
- [Defining Content of Logical Table Sources on page 115](#)

Creating or Removing a Logical Table Source

Use the General tab of the Logical Table Source dialog box to define general properties for the logical table source.

To create a logical table source

- 1 In the Business Model and Mapping layer, right-click a logical table and choose New Object > Logical Table Source.
- 2 In the Logical Table Source dialog box, click the General tab, and then type a name for the logical table source and click Add.

- 3 In the Browse dialog box, you can view joins and select tables for the logical table source.
When there are two or more tables in a logical table source, all of the participating tables must have joins defined between them.
- 4 To view the joins, in the Browse dialog box, select a table and click View.
 - In the Physical Table dialog box, review the joins, and then click Cancel.
- 5 To add tables to the table source, select the tables in the Name list and click Select.
- 6 In the Logical Table Source dialog box, click the Column Mapping tab and complete the fields using the instructions in [“Defining Physical to Logical Table Source Mappings” on page 113](#).
- 7 In the Logical Table dialog box, click the Content tab and complete the fields using the instructions in [“Defining Content of Logical Table Sources” on page 115](#).
- 8 Click OK.

To remove a table as a source

- 1 In the Business Model and Mapping layer, right-click a logical table source and choose Properties.
- 2 In the Logical Table Source dialog box, click the General tab.
- 3 In the tables list, select the table you want to remove and click Remove.
- 4 After removing the appropriate tables, click OK.

Example of Creating Sources for Each Level of Aggregated Fact Data

In addition to creating the source for the aggregate fact table, you should create corresponding logical dimension table sources at the same levels of aggregation.

NOTE: If the sources at each level already exist, you do not need to create new ones. You need to have at least one source at each level referenced in the aggregate content specification.

For example, you might have a monthly sales table containing a precomputed sum of the revenue for each product in each store during each month. You need to have the following three other sources, one for each of the logical dimension tables referenced in the example:

- A source for the Product logical table with one of the following content specifications:
 - By logical level: ProductDimension.ProductLevel
 - By column: Product.Product_Name
- A source for the Store logical table with one of the following content specifications:
 - By logical level: StoreDimension.StoreLevel
 - By column: Store.Store_Name
- A source for the Time logical table with one of the following content specifications:
 - By logical level: TimeDimension.MonthLevel
 - By column: Time.Month

Defining Physical to Logical Table Source Mappings

Use the Column Mapping tab of the Logical Table Source dialog box to map logical columns to physical columns. The physical to logical mapping can also be used to specify transformations that occur between the Physical layer and the Business Model and Mapping layer. The transformations can be simple, such as changing an integer data type to a character, or more complex, such as applying a formula to find a percentage of sales per unit of population.

To map logical columns to physical columns

- 1 In the Business Model and Mapping layer, double-click a logical table source, if the Logical Table Source dialog box is not already open.
- 2 In the Logical Table Source dialog box, click the Column Mapping tab.
- 3 In the Column Mapping tab, maximize or enlarge the dialog box to show all the contents, as shown in [Figure 11 on page 114](#).

NOTE: In the Column Mapping tab, in the Logical column to physical column area, you can sort the rows (toggle among ascending order, descending order, and then restore original order) by clicking a column heading.

- 4 In the Physical Table column, select the table that contains the column you want to map.

When you select a cell in the Physical Table column, a drop-down list appears. It contains a list of tables currently included in this logical table source.

- 5 In the Expression list, select the physical column corresponding to each logical column.

When you select a cell in the Expression column, a drop-down list appears. It contains a list of tables currently included in this logical table source.

- 6 To open the Expression Builder, click the ellipsis button to the left of the Expression you want to view or edit.

NOTE: All columns used in creating physical expressions must be in tables included in the logical table source. You cannot create expressions involving columns in tables outside the source.

- 7 To remove a column mapping, click the delete button next to the Physical Table cell.

You might need to scroll to the right to locate the delete button.

- 8 After you map the appropriate columns, click OK.

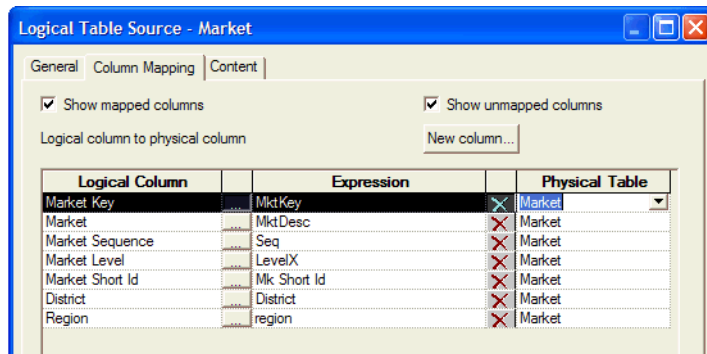


Figure 11. Logical Table Source Dialog Box

To remove a column mapping

- 1 In the Business Model and Mapping layer, right-click a logical table and choose New Object > Logical Table Source.
- 2 In the Logical Table Source dialog box, click the Column Mapping tab.
- 3 In the Column Mapping tab, maximize or enlarge the dialog box to show all the contents, as shown in [Figure 11](#).
- 4 To remove a column mapping, click the delete button next to the Physical Table cell.
- 5 Click OK.

Unmapping a Logical Column from Its Source

In the Logical Column dialog box, the Datatype tab contains information about the logical column. You can edit the logical table sources from which the column derives its data, or unmap it from its sources.

To unmap a logical column from its source

- 1 In the Business Model and Mapping layer, double-click a logical column.
- 2 In the Logical Column dialog box, click the Datatype tab.
- 3 In the Datatype tab, in the Logical Table Source list, select a source and click Unmap.
- 4 Click OK.

Defining Content of Logical Table Sources

To use a source correctly, the Analytics Server has to know what each source contains in terms of the business model. Therefore, you need to define aggregation content for each logical table source of a fact table. The aggregation content rule defines at what level of granularity the data is stored in this fact table. For each dimension that relates to this fact logical table, define the level of granularity, making sure that every related dimension is defined. For more information, see [“Example of Creating Sources for Each Level of Aggregated Fact Data” on page 112](#).

Use the Content tab of the Logical Table Source dialog box to define any aggregate table content definitions, fragmented table definitions for the source, and Where clauses (if you want to limit the number of rows returned).

NOTE: For examples of how to set up fragmentation content for aggregate navigation, see [“Setting Up Fragmentation Content for Aggregate Navigation” on page 227](#).

Verify Joins Exist From Dimension Tables to Fact Table

This source content information tells the Analytics Server what it needs to know to send queries to the appropriate physical aggregate fact tables, joined to and constrained by values in the appropriate physical aggregate dimension tables. Be sure that joins exist between the aggregate fact tables and the aggregate dimension tables in the Physical layer.

One recommended way to accomplish this is to select a fact logical table and request a Business Model Diagram (Selected Tables and Direct Joins). Only the dimension logical tables that are directly joined to this fact logical table appear in the diagram. It does not show dimension tables if the same physical table is used in logical fact and dimension sources.

[Figure 12 on page 115](#) is an example of how the Fact - Asset fact logical table appears in a Business Model Diagram (Selected Tables and Direct Joins) view.

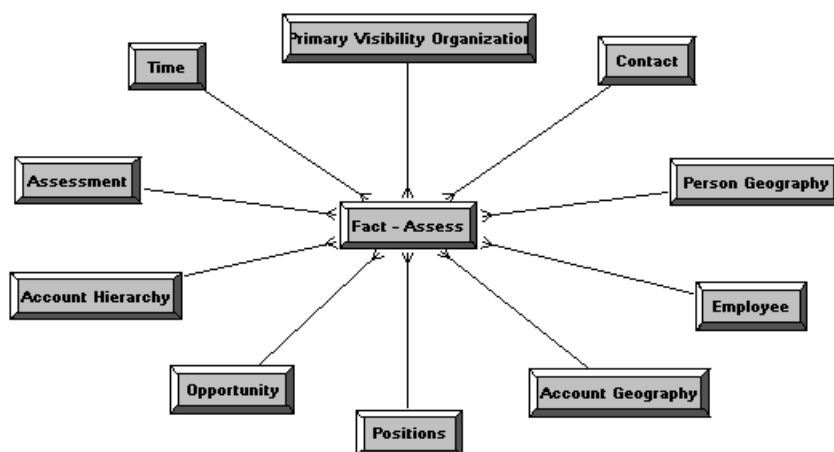


Figure 12. Diagram of Direct Joins for a Fact Table

Table 19 on page 116 contains a list of the logical level for each dimension table that is directly joined the Fact - Assess fact table shown in Figure 12 on page 115.

Table 19. Dimension and Logical Level as Shown in Content Tab

Dimension	Logical Level
Account Geography	Postal Code Detail
Person Geography	Postal Code Detail
Time	Day Detail
Account Organization	Account Detail
Opportunity	Opty Detail
Primary Visibility Organization	Detail
Employee	Detail
Assessment	Detail
Contact (W_PERSON_D)	Detail
FINS Time	Day
Positions	Details

To create logical table source content definitions

- 1 In the Business Model and Mapping layer, double-click a logical table source.
- 2 In the Logical Table Source dialog box, click the Content tab and perform the following steps using Table 20 on page 118 as a guide.
- 3 If a logical source is an aggregate table and you have defined logical dimensions, perform the following steps:
 - a Select Logical Level from the Aggregation content, group-by drop-down list.
CAUTION: Although you have the option to specify aggregate content by logical level or column, it is recommended that you use logical levels exclusively.
 - b In the Logical Level drop-down list, select the appropriate level for each dimension logical table to which the fact logical table is joined.

You should specify a logical level for each dimension, unless you are specifying the Grand Total level. Dimensions with no level specified will be interpreted as being at the Grand Total level.
- 4 If a logical source is an aggregate table and you want to define content for columns, do the following:
 - a Select Column from the Aggregation content, group-by drop-down list.

CAUTION: Although you have the option to specify aggregate content by logical level or column, it is recommended that you use logical levels exclusively.

- b** In the Table pane, select each logical dimension table that defines the aggregation level of the source.
- c** In the Column pane, select the logical column for each dimension that defines how the aggregations were grouped.

When there are multiple logical columns that could be used, select the one that maps to the key of the source physical table. For example, if data has been aggregated to the Region logical level, pick the logical column that maps to the key of the Region table.

NOTE: Do not mix aggregation by logical level and column in the same business model. It is recommended that you use aggregation by logical level.

- 5** To specify fragmented table definitions for the source, use the Fragmentation content window to describe the range of values included in the source when a source represents a portion of the data at a given level of aggregation.

You can type the formula directly into the window, or click the Expression Builder button to the right of the window. In the Fragmentation Content Expression Builder, you can specify content in terms of existing logical columns. For examples of how to set up fragmentation content for aggregate navigation, see [“Specify Fragmentation Content” on page 227](#).

- 6** Select the following option:

This source should be combined with other sources at this level

NOTE: This option is only for multiple sources that are at the same level of aggregation.

- 7** (Optional) Specify Where clause filters in the Where Clause Filter window to limit the number of rows the source uses in the resultant table. For more information, see [“About WHERE Clause Filters” on page 119](#).
 - a** Click the Expression Builder button to open the Physical Where Filter Expression Builder.
 - b** Type the Where clause and click OK.

- 8 Select the option Select distinct values if the values for the source are unique.

Table 20. Content Tab Fields for Logical Table Source

Field	Description
Aggregation content, group by	How the content is aggregated.
More (button)	<p>When you click More, the following options appear:</p> <ul style="list-style-type: none"> ■ Copy. (Available only for fact tables) Copies aggregation content to the Windows clipboard. You can paste the Dimension.Level info into a text editor and use it for searching or for adding to documentation. ■ Copy from. (Available for fact tables and dimension tables) Copies aggregation content from another logical table source in the same business model. You need to specify the source from which to copy the aggregation content. (Multiple business models appear but only the logical table sources from the current business model are selectable.) ■ Get Levels. (Available only for fact tables) Changes aggregation content. If joins do not exist between fact table sources and dimension table sources (for example, if the same physical table is in both sources), the aggregation content determined by the administration tool will not include the aggregation content of this dimension. ■ Check Levels. (Available only for fact tables) check the aggregation content of logical fact table sources (not dimension table sources). The information returned depends on the existence of dimensions and hierarchies with logical levels and level keys, and physical joins between tables in dimension table sources and the tables in the fact table source. (If the same tables exist in the fact and dimension sources and there are no physical joins between tables in the sources, the Check Levels feature will not include the aggregation content of this dimension.)
Fragmentation content	A description of the contents of a data source in business model terms. Data is fragmented when information at the same level of aggregation is split into multiple tables depending on the values of the data. A common situation would be to have data fragmented by time period. For examples of how to set up fragmentation content for aggregate navigation, see “Specify Fragmentation Content” on page 227 .
This source should be combined with other sources at this level (check box)	Check this box when data sources at the same level of aggregation do not contain overlapping information. In this situation, all sources must be combined to get a complete picture of information at this level of aggregation.
Select distinct values	Used if the values for the source are unique.

About WHERE Clause Filters

The WHERE clause filter is used to constrain the physical tables referenced in the logical table source. If there are no constraints on the aggregate source, leave the WHERE clause filter blank.

Each logical table source should contain data at a single intersection of aggregation levels. You would not want to create a source, for example, that had sales data at both the Brand and Manufacturer levels. If the physical tables include data at more than one level, add an appropriate WHERE clause constraint to filter values to a single level.

Any constraints in the WHERE clause filter are made on the physical tables in the source.

About Dimensions and Hierarchical Levels

In a business model, a dimension represents a hierarchical organization of logical columns (attributes) belonging to a single logical dimension table. Common dimensions might be time periods, products, markets, customers, suppliers, promotion conditions, raw materials, manufacturing plants, transportation methods, media types, and time of day. Dimensions exist in the Business Model and Mapping (logical) layer and end users do not see them.

In each dimension, you organize attributes into hierarchical levels. These logical levels represent the organizational rules, and reporting needs required by your business. They provide the structure (metadata) that the Analytics Server uses to drill into and across dimensions to get more detailed views of the data.

Dimension hierarchical levels are used to perform the following actions:

- Set up aggregate navigation
- Configure level-based measure calculations (see [“Level-Based Measure Calculations Example” on page 123](#))
- Determine what attributes appear when Siebel Business Analytics Web users drill down in their data requests

Process of Creating and Administering Dimensions

Each business model can have one or more dimensions, each dimension can have one or more logical levels, and each logical level has one or more attributes (columns) associated with it.

NOTE: The concept of dimensions for a multidimensional data source is less complex than for dimensions in other data sources. For example, you do not create dimension level keys. A dimension is specific to a particular multidimensional data source (cannot be used by more than one) and cannot be created and manipulated individually. Additionally, each cube in the data source should have at least one dimension and one measure in the logical layer.

The following sections explain how to create dimensions:

- [Creating Dimensions](#)

■ [Creating Dimension Levels and Keys on page 120](#)

■ [Creating Dimensions Automatically on page 125](#)

Creating Dimensions

When creating a dimension, each dimension can be associated with attributes (columns) from one or more logical dimension table (plus level-based measures from logical fact tables).

NOTE: The logical column(s) comprising the logical key of a dimension table must be associated with the lowest level of the dimension.

To create a dimension

- 1 In the Business Model and Mapping Layer, right-click a business model and select New Object > Dimension.
- 2 In the Dimension dialog box, type a name for the dimension.
- 3 (Optional) Type a description of the dimension.

NOTE: After you associate a dimension with logical columns, the tables in which these columns exist will appear in the Tables tab.

- 4 Click OK.

Creating Dimension Levels and Keys

A dimension contains two or more logical levels. The recommended sequence for creating logical levels is to create a grand total level and then create child levels, working down to the lowest level. The following are the parts of a dimension:

- **Grand total level.** A special level representing the grand total for a dimension. Each dimension can have just one Grand Total level. A grand total level does not contain dimensional attributes and does not have a level key. However, you can associate measures with a grand total level. The aggregation level for those measures will always be the grand total for the dimension.
- **Level.** All levels, except the Grand Total level, need to have at least one column. However, it is not necessary to explicitly associate all of the columns from a table with logical levels. Any column that you do not associate with a logical level will be automatically associated with the lowest level in the dimension that corresponds to that dimension table. All logical columns in the same dimension table have to be associated with the same dimension.

- **Hierarchy.** In each business model, in the logical levels, you need to establish the hierarchy (parent-child levels). One model might be set up so that weeks roll up into a year. Another model might be set up so that weeks do not roll up. For example, in a model where weeks roll up into a year, it is implied that each week has exactly one year associated with it. This may not hold true for calendar weeks, where the same week could span two years. Some hierarchies might require multiple elements to roll up, as when the combination of month and year roll up into exactly one quarter. You define the hierarchical levels for your particular business so that results from analyses conform to your business needs and requirements.
- **Level keys.** Each logical level (except the topmost level defined as a Grand Total level) needs to have one or more attributes that compose a level key. The level key defines the unique elements in each logical level. The dimension table logical key has to be associated with the lowest level of a dimension and has to be the level key for that level.

A logical level may have more than one level key. When that is the case, specify the key that is the primary key of that level. All dimension sources which have an aggregate content at a specified level need to contain the column that is the primary key of that level. Each logical level should have one level key that will be displayed when a Answers or Intelligence Dashboard user clicks to drill down. This may or may not be the primary key of the level. To set the level key to display, select the Use for drill down check box on the Level Key dialog box.

Be careful using level keys such as Month whose domain includes values January, February, and so on—values that are not unique to a particular month, repeating every year. To define Month as a level key, you also need to include an attribute from a higher level, for example, Year. To add Year, click the Add button in this dialog and select the logical column from the dialog that is presented.

To create and administer dimension hierarchy levels, perform the following tasks:

- [Creating a Logical Level in a Dimension on page 121](#)
- [Associating a Logical Column and Its Table with a Dimension Level on page 122](#)
- [Identifying the Primary Key for a Dimension Level on page 122](#)
- [Adding a Dimension Level to the Preferred Drill Path on page 123](#)
- [Level-Based Measure Calculations Example on page 123](#)
- [Grand Total Dimension Hierarchy Example on page 125](#)

Creating a Logical Level in a Dimension

When creating a logical level in a dimension, you also create the hierarchy by identifying the type of level and defining child levels. For more information about creating hierarchies for a multidimensional data source, see [“Creating the Business Model Layer for a Multidimensional Data Source” on page 104](#).

To define general properties for a logical level in a dimension

- 1 In the Business Model and Mapping layer, right-click a dimension and choose New Object > Logical Level.
- 2 In the Logical Level dialog box, in the General tab, specify a name for the logical level.

- 3 Specify the number of elements that exist at this logical level. If this level will be the Grand Total level, leave this field blank. The system will set to a value of 1 by default.

This number is used by the Analytics Server when picking aggregate sources. The number does not have to be exact, but ratios of numbers from one logical level to another should be accurate.

- 4 If this logical level:
 - Is the grand total level, select the option Grand total level. There should be only one grand total level under a dimension.
 - Rolls up to its parent, select the Supports rollup to parent elements check box.
 - Is not the grand total level and does not roll up, leave both check boxes unselected.

- 5 To define child logical levels, click Add and perform the following steps:

- a Select the child logical levels and click OK to return to the General tab of the Level dialog box.

The child levels appear in the Child Levels pane.

- b To remove a previously defined child level, select the level in the Child Levels pane and click Remove.

The child level and all of its child levels are deleted from the Child Levels pane.

- 6 (Optional) Type a description of the logical level.

- 7 Click OK.

Associating a Logical Column and Its Table with a Dimension Level

After you create all logical levels within a dimension, you need to drag and drop one or more columns from the dimension table to each logical level except the Grand Total level. The first time you drag a column to a dimension it associates the logical table to the dimension. It also associates the logical column with that level of the dimension. To change the logical level to be associated with that logical column, you can drag a column from one logical level to another.

After you associate a logical column with a dimension level, the tables in which these columns exist appear in the Tables tab of the Dimensions dialog box.

To verify tables that are associated with a dimension

- 1 In the Business Model and Mapping layer, double-click a dimension.

- 2 In the Dimensions dialog box, click the Tables tab.

The tables list contains tables that you associated with that dimension. This list of tables includes only one logical dimension table and one or more logical fact tables (if you created level-based measures).

- 3 Click OK or Cancel to close the Dimensions dialog box.

Identifying the Primary Key for a Dimension Level

Use the Keys tab to identify the primary key for a level.

To specify a primary key for a dimension level

- 1 In the Business Model and Mapping layer, expand a dimension and then expand the highest level (grand total level) of the dimension.
- 2 Double-click a logical level below the grand total level.
- 3 In the Logical Level dialog box, click the Keys tab.
- 4 In the Keys tab, from the Primary key drop-down list, select a named level key.
NOTE: If only one level key exists, it will be the primary key by default.
- 5 To add a column to the list, perform the following steps:
 - a In the Logical Level dialog box, click New.
 - b In the Logical Level Key dialog box, type a name for the key.
 - c In the Logical Level Key dialog box, select a column or click Add.
 - d If you click Add, in the Browse dialog box, select the column, and then click OK.
The column you selected appears in the Columns list of the Logical Level Key dialog box and the check box is automatically selected.
- 6 (Optional) Type a description for the key and then click OK.
- 7 Repeat [Step 2 on page 123](#) through [Step 6 on page 123](#) to add primary keys to other logical levels.
- 8 In the Logical Level dialog box, click OK.

Adding a Dimension Level to the Preferred Drill Path

You can use the Preferred Drill Path tab to identify the drill path to use when Siebel Business Analytics Web users drill down in their data requests. You should use this feature only to specify a drill path that is outside the normal drill path defined by the dimensional level hierarchy. This feature is most commonly used to drill from one dimension to another. You can delete a logical level from a drill path or reorder a logical level in the drill path.

To add a dimension level to the preferred drill path

- 1 Click the Add button to open the Browse dialog box, where you can select the logical levels to include in the drill path. You can select logical levels from the current dimension or from other dimensions.
- 2 Click OK to return to the Level dialog box. The names of the levels are added to the Names pane.

Level-Based Measure Calculations Example

A level-based measure is a column whose values are always calculated to a specific level of aggregation. For example, a company might want to measure its revenue based on the country, based on the region, and based on the city. You can set up columns to measure CountryRevenue, RegionRevenue, and CityRevenue. The measure AllProductRevenue is an example of a level-based measure at the Grand Total level.

Creating and Administering the Business Model and Mapping Layer in a Repository

■ Process of Creating and Administering Dimensions

Level-based measures allow a single query to return data at multiple levels of aggregation. They are also useful in creating share measures, that are calculated by taking some measure and dividing it by a level-based measure to calculate a percentage. For example, you can divide salesperson revenue by regional revenue to calculate the share of the regional revenue each salesperson generates.

To set up these calculations, you need to build a dimensional hierarchy in your repository that contains the levels Grandtotal, Country, Region, and City. This hierarchy will contain the metadata that defines a one-to-many relationship between Country and Region and a one-to-many relationship between Region and City. For each country, there are many regions but each region is in only one country. Similarly, for each region, there are many cities but each city is in only one region.

Next, you need to create three logical columns (CountryRevenue, RegionRevenue, and CityRevenue). Each of these columns uses the logical column Revenue as its source. The Revenue column has a default aggregation rule of SUM and has sources in the underlying databases.

You then drag the CountryRevenue, RegionRevenue, and CityRevenue columns into the Country, Region, and City levels, respectively. Each query that requests one of these columns will return the revenue aggregated to its associated level. [Figure 13](#) shows what the business model in the Business Model and Mapping layer would look like for this example.

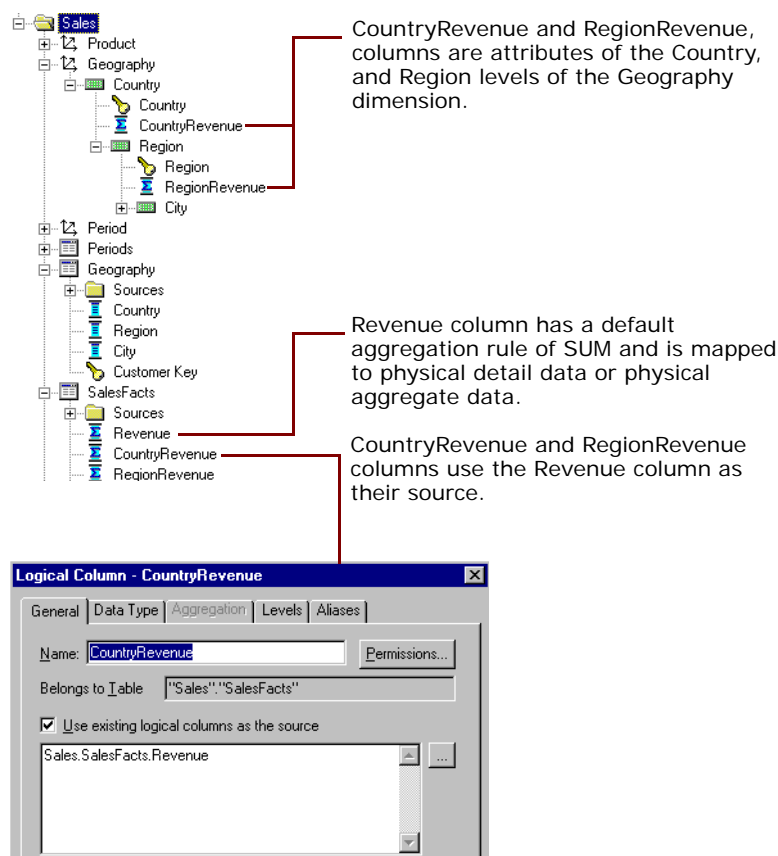


Figure 13. Example Business Model in the Business Model and Mapping Layer

Grand Total Dimension Hierarchy Example

You might have a product dimensional hierarchy with levels TotalProducts (grand total level), Brands, and Products. Additionally, there might be a column called Revenue that is defined with a default aggregation rule of Sum. You can then create a logical column, AllProductRevenue, that uses Revenue as its source (as specified in the General tab of the Logical Column dialog). Now drag AllProductRevenue to the grand total level. Each query that includes this column will return the total revenue for all products. The value is returned regardless of any constraints on Brands or Products. If you have constraints on columns in other tables, the grand total is limited to the scope of the query. For example, if the scope of the query asks for data from 1999 and 2000, the grand total product revenue is for all products sold in 1999 and 2000.

If you have three products, A, B, and C with total revenues of 100, 200, and 300 respectively, then the grand total product revenue is 600 (the sum of each product's revenue). If you have set up a repository as described in this example, the following query produces the results listed:

```
select product, productrevenue, allproductrevenue
from sales_subject_area
where product in 'A' or 'B'
```

PRODUCT	PRODUCTREVENUE	ALLPRODUCTREVENUE
A	100	600
B	200	600

In this example, the AllProductRevenue column will always return a value of 600, regardless of the products the query constrains on.

Creating Dimensions Automatically

You can set up a dimension automatically from a logical dimension table if a dimension for that table does not exist. To create a dimension automatically, the Administration Tool examines the logical table sources and the column mappings in those sources and uses the joins between physical tables in the logical table sources to determine logical levels and level keys. Therefore, it is best to create a dimension in this way after all the logical table sources have been defined for a dimension table.

The following rules apply:

- Create Dimensions is only available if the selected logical table is a dimension table (defined by 1:N logical joins) and no dimension has been associated with this table.
- An automatically created dimension uses the same name as the logical table, adding Dim as a suffix. For example, if a table is named Periods, the dimension is named Periods Dim.
- A grand total level is automatically named *[name of logical table] Total*. For example, the grand total level of the Periods Dim table is Periods Total.
- When there is more than one table in a source, the join relationships between tables in the source determine the physical table containing the lowest level attributes. The lowest level in the hierarchy is named *[name of logical table] Detail*. For example, the lowest level of the periods table is Periods Detail.

- The logical key of the dimension table is mapped to the lowest level of the hierarchy and specified as the level key. This logical column should map to the key column of the lowest level table in the dimension source.
 - If there are two or more physical tables in a source, the columns that map to the keys of those tables become additional logical levels. These additional level names use the logical column names of these key columns.
 - The order of joins determines the hierarchical arrangement of the logical levels. The level keys of these new logical levels are set to the logical columns that map to the keys of the tables in the source.
- If there is more than one logical table source, the tool uses attribute mappings and physical joins to determine the hierarchical order of the tables in the physical sources. For example, you might have three sources (A, B, C) each containing a single physical table and attribute mappings for 10, 15, and 3 attributes, respectively, (not counting columns that are constructed from other logical columns). The following is a list of the results of creating a dimension for this table automatically:
 - The Administration Tool creates a dimension containing 4 logical levels, counting the grand total and detail levels.
 - The key of the table in source B (that has the greatest number of columns mapped and contains the column mapping for the logical table key) should be the level key for the detail level.
 - The parent of the detail level should be the logical level named for the logical column that maps to the key of the physical table in source A.
 - Any attributes that are mapped to both A and B should be associated with level A.
 - The parent of level A should be the logical level named for the logical column that maps to the key of the physical table in source C.
 - Any columns that are mapped to both A and C should be associated with level C.
- Table joins in a physical source might represent a pattern that results in a split hierarchy. For example, the Product table may join to the Flavor table and a Subtype table. This would result in two parents of the product detail level, one flavor level and one subtype level.
- You cannot create a dimension automatically in the following situations:
 - If a dimension with joins and levels has already been created, Create Dimension will not appear on the right-click menu.
 - If the table is not yet joined to any other table, the option is not available because it is considered a fact table.
- In a snowflake schema, if you use a table with only one source, and create the dimension automatically the child tables will automatically be incorporated into a hierarchy. The child tables will form intermediate levels between the grand total level and detail level. If more than one child table exists for a dimension table, while creating dimension automatically hierarchy will be split hierarchy.

To create a dimension automatically

- 1 In the Administration Tool, open a repository.
- 2 In the Business Model and Mapping layer of a repository, right-click a logical table.
- 3 From the right-click menu, choose Create Dimension.

A dimension appears in the Business Model and Mapping layer.

Setting Up Dimension-Specific Aggregate Rules for Logical Columns

The majority of measures have the same aggregation rule for each dimension. However, some measures can have different aggregation rules for different dimensions. For example, bank balances might be averaged over time but summed over the individual accounts. The Analytics Server allows you to configure dimension-specific aggregation rules. You can specify one aggregation rule for a given dimension and specify other rules to apply to other dimensions.

You need to configure dimensions in the Business Model and Mapping layer to set up dimension-specific aggregation. For more information about setting up aggregate navigation, see [“Setting Up Fragmentation Content for Aggregate Navigation” on page 227](#).

To specify dimension-specific aggregation rules for a single logical column

- 1 In the Business Model and Mapping layer, double-click a logical column.
 - 2 In the Logical Column dialog box, click the Aggregation tab.
 - 3 In the Aggregation tab, select the Based on dimensions check box.
The Browse dialog box automatically opens.
 - 4 In the Browse dialog box, select a dimension, and then click OK.
 - 5 Click New, select a dimension over which you want to aggregate, and then click OK.
 - 6 From the Formula drop-down list, select a rule, and then click OK.
- NOTE:** After selecting rules for specified dimensions, set the aggregation rule for any remaining dimensions by using the dimension labeled Other.
- 7 If you need to create more complex formulas, click the ellipsis button to the right of the Formula column to open the Expression Builder - Aggregate.
 - 8 To change the order in which the dimension-specific rules are performed, click Up or Down, and then click OK.

When calculating the measure, aggregation rules are applied in the order (top to bottom) established in the dialog box.

To specify dimension-specific aggregation rules for multiple logical fact columns

- 1 In the Business Model and Mapping layer, select multiple logical fact columns.

- 2 Right-click and select Set Aggregation.

If the fact column has an aggregation rule, Set Aggregation will not appear in the menu.

- 3 In the Aggregation dialog box, select or clear the All columns the same check box.

The check box is checked by default. When checked, you can set aggregation rules that will apply to all selected columns. If you clear the check box, you can set aggregation rules separately for each selected column.

- 4 In the Aggregation tab, click the Use advanced options check box.

- 5 In the Browse dialog box, select a dimension over which you want to perform aggregation, and then click OK.

After setting up the rule for a dimension, specify aggregation rules for any other dimensions in the entry labeled Other.

- 6 Click the ellipsis button to the right of the Formula column.

- 7 In the Expression Builder - Aggregate dialog box, from the Formula drop-down list, select the aggregation to perform over the dimension.

- 8 To change the order in which the dimension-specific rules are performed, click Up or Down, and then click OK.

When calculating the measure, aggregation rules are applied in the order (top to bottom) established in the dialog box.

Setting Up Display Folders in the Business Model and Mapping Layer

Administrators can create display folders to organize objects in the Business Model and Mapping layer. They have no metadata meaning. After you create a display folder, the selected tables and dimensions appear in the folder as a shortcut and in the business model tree as the object. You can hide the objects so that you only see the shortcuts in the display folder. For more information about hiding these objects, see [“Using the Options Dialog Box—Repository Tab” on page 34](#).

NOTE: Deleting objects in the display folder only deletes the shortcuts to those objects.

To set up a logical display folder

- 1 In the Business Model and Mapping layer, right-click a business model, and choose New Object > Logical Display Folder.
- 2 In the Logical Display Folder dialog box, in the Tables tab, type a name for the folder.
- 3 To add tables to the display folder, perform the following steps:
 - a Click Add.
 - b In the Browse dialog box, select the fact or dimension tables you want to add to the folder and click Select.
- 4 To add dimensions to the display folder, click the Dimensions tab and perform the following steps:

- a Click Add.
 - b In the Browse dialog box, select the dimensions that you want to add to the folder and click Select.
- 5 Click OK.

Defining Logical Joins

Logical tables are related to each other. How they are related is expressed in logical joins. A key property of a logical join is cardinality. Cardinality expresses how rows in one table are related to rows in the table to which it is joined. A one-to-many cardinality means that for every row in the first logical dimension table there are 0, 1, or many rows in the second logical table. The Administration Tool considers a table to be a logical fact table if it is at the Many end of all logical joins that connect it to other logical tables.

Specifying the logical table joins is required so that the Analytics Server can have the necessary metadata to translate a logical request against the business model to SQL queries against the physical data sources. The logical join information provides the Analytics Server with the many-to-one relationships between the logical tables. This logical join information is used when the Analytics Server generates queries against the underlying databases.

The joins between the logical layer and the physical layer will be automatically created if both of the following statements are true:

- You create the logical tables by simultaneously dragging and dropping all required physical tables to the Business Model and Mapping layer.
- The logical joins are the same as the joins in the Physical layer.

However, you will probably have to create some logical joins in the Business Model and Mapping layer, because you will rarely drag and drop all physical tables simultaneously except in very simple models. In the Business Model and Mapping layer, you should create complex joins with one-to-many relationships and not key or foreign key joins.

You can create logical foreign keys and logical complex joins using either the Joins Manager or the Business Model Diagram. When you create a complex join in the Physical layer, you can specify expressions and the specific columns on which to create the join. When you create a complex join in the Business Model and Mapping layer, you cannot specify expressions or columns on which to create the join. The existence of a join in the Physical layer does not require a matching join in the Business Model and Mapping layer.

CAUTION: It is recommended that you do not have foreign keys for logical tables. However, you can create logical foreign keys and logical complex joins using either the Joins Manager or the Business Model Diagram. A logical key for a fact table must be made up of the key columns that join to the attribute tables.

To create logical joins, perform the following tasks:

- [Defining Logical Joins with the Joins Manager on page 130](#)
- [Defining Logical Joins with the Business Model Diagram on page 131](#)

Defining Logical Joins with the Joins Manager

You can use the Joins Manager to view logical join relationships and to create logical foreign keys and complex joins.

This section includes the following topics:

- [Creating a Logical Foreign Key on page 130](#)
- [Creating a Logical Complex Join on page 131](#)

Creating a Logical Foreign Key

Typically, you should not create logical foreign keys. This capability is in the Administration Tool to provide compatibility with previous releases. Logical foreign key joins might be needed if the Analytics Server is to be used as an ODBC data source for certain third-party query and reporting tools.

To create a logical foreign key

- 1 In the Administration Tool toolbar, select Manage > Joins.
The Joins Manager dialog box appears.
- 2 Select Action > New > Logical Foreign Key
- 3 In the Browse dialog box, double-click a table.
The Logical Foreign Key dialog box appears.
- 4 Type a name for the foreign key.
- 5 In the Table drop-down list on the left side of the dialog box, select the table that the foreign key references.
- 6 Select the columns in the left table that the foreign key references.
- 7 Select the columns in the right table that make up the foreign key columns.
- 8 (Optional) To specify a driving table for the key, select a table from the Driving drop-down list, and an applicable cardinality.

This is for use in optimizing the manner in which the Analytics Server processes multi-database inner joins when one table is very small and the other table is very large. Do not select a driving table unless multi-database joins are going to occur. For more information about driving tables, see [“Specifying a Driving Table” on page 132](#).

CAUTION: Use extreme caution in deciding whether to specify a driving table. Driving tables are used for query optimization only under rare circumstances and when the driving table is extremely small, that is, less than 1000 rows. Choosing a driving table incorrectly can lead to severe performance degradation.

- 9 Select the join type from the Type drop-down list.
- 10 To open the Expression Builder, click the button to the right of the Expression pane.
The expression displays in the Expression pane.

- 11 Click OK to save your work.

Creating a Logical Complex Join

The use of logical complex joins is recommended over logical key and foreign key joins.

To create a logical complex join

- 1 In the Administration Tool toolbar, select Manage > Joins.
The Joins Manager dialog box appears.
- 2 Select Action > New > Logical Complex Join.
The Logical Join dialog box appears.
- 3 Type a name for the complex join.
- 4 In the Table drop-down lists on the left and right side of the dialog box, select the tables that the complex join references.
- 5 (Optional) To specify a driving table for the key, select a table from the Driving drop-down list, and an applicable cardinality.

This is for use in optimizing the manner in which the Analytics Server processes multi-database inner joins when one table is very small and the other table is very large. Do not select a driving table unless multi-database joins are going to occur. For more information about driving tables, see [“Specifying a Driving Table” on page 132](#).

CAUTION: Use extreme caution in deciding whether to specify a driving table. Driving tables are used for query optimization only under rare circumstances and when the driving table is extremely small, that is, less than 1000 rows. Choosing a driving table incorrectly can lead to severe performance degradation.

- 6 Select the join type from the Type drop-down list.
- 7 Click OK.

Defining Logical Joins with the Business Model Diagram

The Business Model Diagram shows logical tables and any defined joins between them.

NOTE: It is recommended that you use the Business Model Diagram to define complex joins. It is recommended that you do not use the Diagram to define logical foreign keys. Logical foreign key joins may be needed if the Analytics Server is to be used as an ODBC data source for certain third-party query and reporting tools.

To display the Business Model Diagram

- 1 In the Administration Tool, right-click a business model, and then select Business Model Diagram > Whole Diagram.
- 2 Click one of the following buttons on the Administration Tool toolbar:

- New complex join (Recommended)
- New foreign key (Not recommended. This capability exists to provide compatibility with previous releases.)

- 3 With this button selected, move the cursor to the first table in the join (the one of the one-to-many join).
- 4 Left-click and move the cursor to the table to which you want to make the join (the many of the one-to-many join), and then left-click on the second table.

The Logical Foreign Key or Logical Join dialog box appears.

- 5 For a logical foreign key, select the joining columns from the left and the right tables.
- 6 (Optional) To specify a driving table for the key, select a table from the Driving drop-down list, and an applicable cardinality.

This is for use in optimizing the manner in which the Analytics Server processes multi-database inner joins when one table is very small and the other table is very large. Do not select a driving table unless multi-database joins are going to occur. For more information about driving tables, see [“Specifying a Driving Table” on page 132](#).

CAUTION: Use extreme caution in deciding whether to specify a driving table. Driving tables are used for query optimization only under rare circumstances and when the driving table is extremely small, that is, less than 1000 rows. Choosing a driving table incorrectly can lead to severe performance degradation.

- 7 Select the join type from the Type drop-down list.
- 8 To open the Expression Builder, click the button to the right of the Expression pane.
Only columns, designated predicates, and operators are allowed in the expression.
- 9 Click OK to save your work.

Specifying a Driving Table

You can specify a driving table for logical joins from the Logical Joins window. Driving tables are for use in optimizing the manner in which the Analytics Server processes cross-database joins when one table is very small and the other table is very large. Specifying driving tables leads to query optimization only when the number of rows being selected from the driving table is much smaller than the number of rows in the table to which it is being joined.

CAUTION: To avoid problems, only specify driving tables when the driving table is extremely small - less than 1000 rows.

When you specify a driving table, Analytics Server will use it if the query plan determines that its use will optimize query processing. The small table (the driving table) is scanned, and parameterized queries are issued to the large table to select matching rows. The other tables, including other driving tables, are then joined together.

CAUTION: If large numbers of rows are being selected from the driving table, specifying a driving table could lead to significant performance degradation or, if the MAX_QUERIES_PER_DRIVE_JOIN limit is exceeded, query termination.

In general, driving tables can be used with inner joins, and for outer joins when the driving table is the left table for a left outer join, or the right table for a right outer join. Driving tables are not used for full outer joins. For instructions about specifying a driving table, see ["Defining Logical Joins" on page 129](#).

There are two entries in the database features table that control and tune drive table performance.

■ **MAX_PARAMETERS__PER_DRIVE_JOIN**

This is a performance tuning parameter. In general, the larger its value, the fewer parameterized queries that will need to be generated. Values that are too large can result in parameterized queries that fail due to back-end database limitations. Setting the value to 0 (zero) turns off the drive table joins feature.

■ **MAX_QUERIES_PER_DRIVE_JOIN**

This is used to prevent runaway drive table joins. If the number of parameterized queries exceeds its value, the query is terminated and an error message is returned to the user.

Identifying Physical Tables That Map to Logical Objects

The Physical Diagram shows the physical tables that map to the selected logical object and the physical joins between each table.

One of the joins options, Object(s) and Direct Joins within Business Model, is unique to the logical layer. It creates a physical diagram of the tables that meet both of the following conditions:

- Tables in the selected objects and tables that join directly
- Tables that are mapped (exist in logical table sources in the business model) in the business model

To open the physical diagram of a logical object

- 1 In the Business Model and Mapping layer, right-click a business model, logical table, or logical table source.
- 2 Choose Physical Diagram and then one of the joins options.
- 3 Click and drag any object to more clearly see the relationship lines such as one-to-many.

6

Creating and Maintaining the Presentation Layer in a Repository

This section is part of the roadmap for planning and setting up a repository. For more information, see [“Planning and Creating a Siebel Business Analytics Repository” on page 41](#). After you have created the Business Model and Mapping layer, you can drag and drop that layer to the Presentation layer in the Administration Tool. For more information about the Business Model and Mapping layer, see [“Creating and Administering the Business Model and Mapping Layer in a Repository” on page 103](#).

This section provides instructions for using the Administration Tool to create and edit objects in the Presentation layer of a repository. This is the fourth step in setting up a repository.

This chapter contains the following topics:

- [Creating the Presentation Layer in the Repository on page 135](#)
- [Presentation Layer Objects on page 136](#)
- [Generating an XML File from a Presentation Table on page 141](#)

Creating the Presentation Layer in the Repository

The Presentation layer provides a way to present customized views of a business model (known as *Presentation Catalogs*) to users. Presentation Catalogs in the Presentation layer are seen as business models by Siebel Business Analytics Web users. They appear as catalogs to client tools that use the Analytics Server as an ODBC data source. The following topics describe the Process of creating the Presentation layer.

NOTE: In offline editing, remember to save your repository from time to time. You can save a repository in offline mode even though the business models may be inconsistent.

Copy Business Models to Publish to Users

There are several ways to create a Presentation Catalog in the Presentation layer. The recommended method is to drag and drop a business model from the Business Model and Mapping layer to the Presentation layer, and then modify the Presentation layer based on what you want users to see. You can move columns between presentation tables, remove columns that do not need to be seen by the users, or even present all of the data in a single presentation table. You can create presentation tables to organize and categorize measures in a way that makes sense to your users.

Remove Any Unneeded or Unwanted Columns

One important reason to use a custom Presentation layer is to make the schema as easy to use and understand as possible. Therefore, users should not see columns that have no meaning to them. The following columns are examples of columns that you might want to remove from the Presentation layer:

- Key columns that have no business meaning.
- Columns that users do not need to see (for example, codes, when text descriptions exist).
- Columns that users are not authorized to see.

NOTE: You can also restrict access to tables or columns in the security layer. For more information, see [Chapter 16, "Security in Siebel Business Analytics."](#)

Rename Presentation Columns to User-Friendly Names

By default, presentation columns have the same name as the corresponding logical column in the Business Model and Mapping layer. However, you can specify a different name to be shown to users by changing the name in the Presentation Column dialog box. Whenever you change the name of a presentation column, an alias is automatically created for the old name, so compatibility to the old name remains.

Export Logical Keys in the Presentation Catalog

For each presentation catalog in the Presentation layer, decide whether to export any logical keys as key columns to tools that access it in the Presentation Catalog dialog box. Exporting logical keys is irrelevant to users of Siebel Business Analytics Web, but it may be advantageous for some query and reporting tools. If you decide to export logical keys, be sure the logical key columns exist in the table folders. In this situation, your business model should use logical key/foreign key joins.

When you select the option Export logical keys in the Presentation Catalog dialog box, any columns in the Presentation layer that are key columns in the Business Model and Mapping layer are listed as key columns to any ODBC client. This is the default selection. In most situations, this option should be selected.

NOTE: If you are using a tool that issues parameterized SQL queries, such as Microsoft Access, do not select the option Export logical keys. Not exporting logical keys stops the tool from issuing parameterized queries.

Presentation Layer Objects

The Presentation layer adds a level of abstraction over the Business Model and Mapping layer. It is the view of the data seen by client tools and applications.

The Presentation layer provides a means to further simplify or customize the Business Model and Mapping layer for end users. For example, you can hide key columns or present the schema as a single table. Simplifying the view of the data for users makes it easier to craft queries based on users' business needs.

The section provides instructions for using the Administration Tool's Presentation layer dialog boxes to create and edit repository objects.

This section includes the following topics:

- [Working with Presentation Catalogs on page 137](#)
- [Working with Presentation Tables on page 138](#)

- [Working with Presentation Columns on page 139](#)
- [Using the Alias Tab of Presentation Layer Dialog Boxes on page 141](#)

Working with Presentation Catalogs

In the Presentation layer, presentation catalogs allow you to show different views of a business model to different sets of users. Presentation catalogs have to be populated with contents from a single business model. They cannot span business models.

When creating a presentation catalog, selecting the option Export logical keys causes any columns in the Presentation layer that are key columns in the Business Model and Mapping layer to be listed as key columns to any ODBC client. This is the default selection. In most situations, this option should be selected. Many client tools differentiate between key and nonkey columns, and the option Export logical keys provides client tools access to the key column metadata. Any join conditions the client tool adds to the query, however, are ignored; the Analytics Server uses the joins defined in the repository.

If you set an implicit fact column this column will be added to a query when it contains columns from two or more dimension tables and no measures. You will not see the column in the results. It is used to specify a default join path between dimension tables when there are several possible alternatives.

The Presentation Catalog dialog box has three tabs: General, Presentation Tables, and Aliases. The functionality provided in each tab is described in [Table 21](#).

Table 21. Presentation Catalog Dialog Box

Tab	Comment
General	Use this tab to create or edit a presentation catalog.
Presentation Table	Use this tab to reorder or sort the Presentation layer tables in the Administration Tool workspace, and to delete tables. You can also use this tab to access the Presentation Table dialog box, where you can create and edit tables.
Aliases	Use this tab to specify or delete an alias for a catalog folder.

To create a presentation catalog

- 1 In the Presentation layer, right-click and select New Presentation Catalog.
- 2 In the Presentation Catalog dialog box, in the General tab, type a name for the presentation catalog and click Permissions.
- 3 In the Permissions dialog box, assign user or group permissions to the catalog folder, and then click OK.

For more information about assigning permissions to a presentation catalog, see [“Setting Permissions for Repository Objects” on page 35](#).

- 4 In the Presentation Catalog dialog box, from the Business Model drop-down list, select a business model.

After you add columns to the Presentation Catalog, the drop-down list becomes inactive.

- 5 To expose the logical keys to other applications, select the option Export logical keys.

NOTE: If you are using a tool that issues parameterized SQL queries, such as Microsoft Access, do not select the Export logical keys option. Not exporting logical keys stops the tool from issuing parameterized queries.

- 6 (Optional) Type a description of the catalog folder.

CAUTION: When you move columns into presentation catalog folders, be sure columns with the same name or an alias of the same name do not already exist in the catalog.

- 7 Set an Implicit Fact Column.

- 8 Click OK.

Working with Presentation Tables

You can use presentation tables to organize columns into categories that make sense to the user community. Presentation tables in the Presentation layer contain columns. A presentation table can contain columns from one or more logical tables. The names and object properties of the presentation tables are independent of the logical table properties.

The Presentation Tables dialog box has three tabs: General, Columns, and Aliases. The functionality provided in each tab is described in [Table 22](#).

Table 22. Presentation Tables Dialog Box

Tab	Comment
General	Use this tab to create or edit a presentation table.
Columns	Use this tab to reorder or sort the Presentation layer columns in the Administration Tool workspace, and to delete columns. You can also use this tab to access the Presentation Column dialog box, where you can create and edit columns.
Aliases	Use this tab to specify or delete an alias for a presentation table.

To create a presentation table

- 1 Right-click a catalog folder in the Presentation layer, and then select New Presentation Table from the shortcut menu.

The Presentation Table dialog box appears.

- 2 In the General tab, specify a name for the table.

- 3 Click the Permissions button to open the Permissions dialog box, where you can assign user or group permissions to the table.

For more information about assigning permissions to a presentation table, see [“Setting Permissions for Repository Objects” on page 35](#).

- 4 (Optional) Type a description of the table.

NOTE: To give the appearance of nested folders in Answers, prefix the name of the presentation folder to be nested with a hyphen and a space and place it after the folder in which it nests (- <folder name>). For example, to nest the Sales Facts folder in the Facts folder, place the Sales Facts folder directly after Facts in the metadata and change its name to - Sales Facts. When Answers displays the folder name in the left pane, it omits the hyphen and space from the folder name. To nest a second folder, for example Marketing Facts, in the Facts folder, change its name to - Marketing Facts and place it directly after Sales Facts. The standard preconfigured repositories provide additional examples for you to review.

To delete a presentation table

- 1 In the Presentation layer, right-click a catalog and select Properties.
- 2 In the Presentation Catalog dialog box, click the Presentation Tables tab.
- 3 In the Presentation Tables tab, select a table and click Remove.
A confirmation message appears.
- 4 Click Yes to remove the table, or No to leave the table in the catalog.
- 5 Click OK.

To reorder a table or sort all tables in a presentation catalog

- 1 In the Presentation layer, right-click a catalog and select Properties.
- 2 In the Presentation Catalog dialog box, click the Presentation Tables tab.
- 3 To move a table, perform the following steps:
 - a In the Presentation Tables tab, in the Name list, select the table you want to reorder.
 - b Use drag-and-drop to reposition the table, or click the Up and Down buttons.
- 4 To sort all tables in alphanumeric order, click the Name column heading.

This toggles the sort between ascending and descending alphanumeric order.

Working with Presentation Columns

The presentation column names are, by default, identical to the logical column names in the Business Model and Mapping layer. However, you can present a different name by clearing both the Use Logical Column Name and the Display Custom Name check boxes in the Presentation Column dialog box.

To provide a convenient organization for your end users, you can drag and drop a column from a single logical table in the Business Model and Mapping layer onto multiple presentation tables. This allows you to create categories that make sense to the users. For example, you can create several presentation tables that contain different classes of measures—one containing volume measures, one containing share measures, one containing measures from a year ago, and so on.

The Presentation Column dialog box has the following tabs:

- **General.** Use this tab to create or edit presentation columns.
- **Aliases.** Use this tab to specify or delete an alias for a presentation column.

To create a presentation column

- 1 Right-click a presentation table in the Presentation layer, and then choose New Presentation Column.
- 2 In the Presentation Column dialog box, to use the name of the logical column for the presentation column, select the Use Logical Column check box.

The name of the column and its associated path in the Business Model and Mapping layer appears in the Logical Column Name field.

- 3 To specify a name that is different from the Logical Column name, clear the Use Logical Column check box, and then type a name for the column.
- 4 To assign user or group permissions to the column, click Permissions.
- 5 In the Permissions dialog box, assign permissions, and then click OK.

For more information about assigning permissions, see [“Setting Permissions for Repository Objects” on page 35](#).

- 6 To select the logical column, click Browse.
- 7 In the Browse dialog box, select the column, and then click Select.
- 8 (Optional) Type a description of the presentation column.
- 9 To define any aliases for the logical column, click the Aliases tab.

To edit a presentation column

- 1 In the Business Model and Mapping layer, double-click the presentation column.
- 2 In the Presentation Column dialog box, click Edit.
- 3 In the Logical Column dialog box, make any changes or review the information, and then click OK.

To delete a presentation column

- 1 Right-click a presentation table in the Presentation layer, and then select Properties.
- 2 Click the Columns tab.

- 3 Select the column you want to delete.
- 4 Click Remove, or press the Delete key.

To reorder a presentation column

- 1 Right-click a presentation table in the Presentation layer, and then select Properties.
- 2 Click the Columns tab.
- 3 Select the column you want to reorder.
- 4 Use the drag-and-drop feature to reposition the column, or click the Up and Down buttons.

Using the Alias Tab of Presentation Layer Dialog Boxes

An Alias tab appears on the Presentation Catalog, Presentation Table, and Presentation Column dialog boxes. You can use this tab to specify or delete an alias for the Presentation layer objects.

To add an alias

- 1 Double-click a presentation catalog.
- 2 In the Presentation Layer dialog box, click the Aliases tab.
- 3 Click the new button, and then type the text string to use for the alias.
- 4 Click OK.

To delete an alias

- 1 Double-click a presentation catalog.
- 2 In the Presentation Layer dialog box, click the Aliases tab.
- 3 In the Aliases list, select the alias you want to delete.
- 4 Click the delete button, and then click OK.

Generating an XML File from a Presentation Table

You can import the structure of an analytics presentation table (folder) into Oracle's Siebel Tools. To do this, you create an XML file from a table in the Presentation layer of an analytics repository and then import the XML file into Siebel Tools.

For more information, see the chapter about external business components in *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II*.

7

Completing Setup and Managing Repository Files

This section is part of the roadmap for planning and setting up a repository. For more information, see [“Planning and Creating a Siebel Business Analytics Repository” on page 41](#). After you have created the repository file, the Physical layer, Business Model and Mapping layer, and Presentation layer, you need to perform several tasks to complete the initial repository setup. This section contains these setup steps and topics for managing your repository files.

This section contains instructions for the following topics:

- [Process of Completing the Setup for a Repository File](#)
- [Importing from Another Repository on page 146](#)
- [Querying and Managing Repository Metadata on page 147](#)
- [Constructing a Filter for Query Results on page 150](#)
- [Comparing Repositories on page 151](#)
- [Merging Analytics Repositories on page 153](#)
- [Exporting Analytics Metadata to IBM DB2 Cube Views on page 157](#)
- [Creating an Analytics Multiuser Development Environment on page 157](#)
- [Setting Up the Repository to Work with Delivers on page 166](#)

Process of Completing the Setup for a Repository File

Perform the following tasks to complete the repository file setup:

- [Saving the Repository and Checking Consistency on page 143](#)
- [Add an Entry in the NQConfig.INI File on page 144](#)
- [Create the Data Source on page 145](#)
- [Start the Analytics Server on page 145](#)
- [Test and Refine the Repository on page 145](#)
- [Publish to User Community on page 146](#)

Saving the Repository and Checking Consistency

In offline editing, remember to save your repository from time to time. You can save a repository in offline mode even though the business models may be inconsistent.

To determine if business models are consistent, use the Check Consistency command to check for compilation errors. You can check for errors in the whole repository with the File > Check Global Consistency command or in a particular logical business model by selecting a business model and using the Check Consistency command from the right-click menu.

The consistency check analyzes the repository for certain kinds of errors and inconsistencies. For example, the consistency check finds any logical tables that do not have logical sources configured or any logical columns that are not mapped to physical sources, checks for undefined logical join conditions, determines whether any physical tables referenced in a business model are not joined to the other tables referenced in the business model, and checks for existence of a presentation catalog for each business model. Passing a consistency check does not guarantee that a business model is constructed correctly, but it does rule out many common problems.

When you check for consistency, any errors or warnings that occur are displayed in a dialog box. Correct any errors and check for consistency again, repeating this process until there are no more errors. An error message indicates a problem that needs to be corrected. A warning message identifies a possible problem to the administrator. See [“Checking the Consistency of a Repository or a Business Model” on page 30](#).

NOTE: The consistency check algorithm has been enhanced for Siebel Business Analytics 7.8.2. After upgrading from a previous Siebel Business Analytics software version and checking the consistency of your repository, you might see messages that you had not received in previous consistency checks. This typically indicates inconsistencies that had been undetected prior to the upgrade, not new errors.

Add an Entry in the NQSConfig.INI File

After you build a repository and it is consistent, you need to add an entry in the NQSConfig.INI file for the repository. The entry allows the Analytics Server to load the repository into memory upon startup. The NQSConfig.INI file is located in the following location:

[drive path]:\Siebel Analytics\Config\

To add an entry in the NQSConfig.INI file

- 1 Open the NQSConfig.INI file in an editor such as Notepad.
- 2 In the repository section of file, add an entry for your new repository in this format:

logical_name = repository_file_name ;

For example, if the repository file is named northwind.rpd and the logical name you assign it is star, the entry will read as follows:

star = northwind.rpd ;

One of the repositories should be specified as the default and using the same repository name, the entry would read as follows:


```
star = northwind.rpd, default t;
```

The logical name is the name end users have to configure when configuring a DSN in the Siebel Business Analytics ODBC setup wizard. Filenames consisting of more than a single word should be enclosed in single quotes. Save the configuration file after adding the entry. For more information about the NQSCfg.INI file, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

- 3 Save the file.

Create the Data Source

For end user client applications to connect to the new repository, each user needs to define a data source using an ODBC driver.

NOTE: Siebel Business Analytics Web has the same relationship to the Analytics Server as any other client application.

The steps to create a new data source are given in [Chapter 13, "Connectivity and Third-Party Tools."](#) You can create standard data sources, and data sources that will participate in a cluster.

Start the Analytics Server

When you start the Analytics Server, the repositories specified in the NQSCfg.INI file are loaded and become available for querying. For detailed information on starting the server, see ["Starting the Analytics Server \(Siebel Analytics Server Service\)" on page 237](#).

Test and Refine the Repository

When the repository is created and you can connect to it, run sample queries against it to test that it is created properly. Correct any problems you find and test again, repeating this process until you are satisfied with the results.

Tips For Performance Tuning

The Physical Data Model should more closely resemble the Siebel Business Analytics metadata model (for example the star schema) instead of an transactional database system (approximately 3NF). If the Physical model is set up like the underlying transactional model, performance problems and configuration problems could likely arise. For additional tips, see ["Guidelines For Designing a Repository" on page 51](#).

NOTE: Make sure the metadata is generating the correct record set first, then focus on performance tuning activities (such as adding sources).

- Accuracy of metadata is more important than improved performance.
- In general, push as much processing to the database as possible. This includes tasks such as filtering, string manipulation, and additive measures.

- Move as much of the query logic to the ETL as possible to improve system response time. Pre-calculation of additive metrics and attributes will reduce query complexity and therefore response time.
- Use base and extension tables to allow for a cleaner upgrade path. To improve runtime performance, merge the two tables into a third one, which is then mapped into Siebel Business Analytics. Although this technique requires more effort and a larger ETL batch window, it insulates the system from upgrade errors while still providing optimal performance.
- Denormalize data into _DX tables using the ETL process to reduce runtime joins to other tables.

Publish to User Community

After testing is complete, notify the user community that the data sources are available for querying. Web users need only know the URL to type in their browser. Client/server users (for example, users accessing the Analytics Server with a query tool or report writer client application) need to know the subject area names, the machine on which the server is running, their user IDs and passwords, and they need to have the ODBC setup installed on their PCs. They may also need to know the logical names of repositories when multiple repositories are used and the data source name (DSN) being created does not point to the default repository.

Importing from Another Repository

Use the Repository Import Wizard to import a presentation catalog and its associated children business model and physical layer objects from another repository. You can also import users, groups, variables, initialization blocks, and projects. Use this feature when the objects you import are unrelated to objects already in the repository such as when the business model and physical layer objects do not exist. If an object of the same name and type exists, the import process overwrites the existing object with the new object. When you import objects from one repository into another, the repository from which you are importing must be consistent.

NOTE: This option is not available when opening a repository online.

To import from another repository

- 1 In the Administration Tool, open the repository in which you want to import objects, and then choose File > Import from Repository.

This option is not available when opening a repository online.

- 2 In the Repository Import Wizard, select a repository by its file name or, if it is being used by Siebel Business Analytics, by the ODBC DSN that points to the desired repository on that server, and then click Next.
- 3 Type the Analytics Server administrator user ID and password.

- 4 In the Repository Import Wizard-Objects to Update dialog box, from the drop-down list, choose a category using [Table 23 on page 147](#) as a guide.

Available buttons (options) depend on the category that you select. You can add only the selected object, the selected object with its child objects, or the selected object with its parent objects.

- 5 Repeat [Step 4](#) until you have added all the objects you want to import.

If any objects need to be checked out, the Check Out Objects screen appears, notifying you that objects will be checked out.

- 6 Click Next to continue.

- 7 In the Finish screen, click finish.

If the Administration Tool is open in online mode, you will be prompted to check in the changes before closing the repository.

Table 23. Categories of Repository Objects to Import

Category	Description
Catalogs	When you select a catalog, the Add with Children button become active. Presentation catalogs are always added with all their child objects. All associated objects, from the Presentation layer to the Physical layer, will be updated and synchronized.
Groups	When you select a group, the Add, Add with Children, and Add with Parents buttons become active. (You can view information about group membership from the Security Manager.)
Initialization Blocks	When you select an initialization block, the Add with Children button becomes active.
List Catalogs	When you select a list catalog, the Add with Children button becomes active.
Projects	When you select a project, all repository objects that you choose to update and synchronize appear.
Target levels	When you select a target level, the Add with Children button is active.
Users	When you select a user, the buttons that become active depend on the user that you select in the left pane.
Variables	When you select a variable, the Add and Add with Parents buttons are active. Defined system and session variables appear in the left pane.

Querying and Managing Repository Metadata

You can query for objects in the repository using the Query Repository tool. If you query using the All Types option, you see a listing of the exposed object types in the repository. The list does not contain objects such as aggregate rules, logical source folders, privilege packages, and other objects that are considered internal objects.

You can use repository queries to help manage the repository metadata in the following ways:

- Examine and update the internal structure of the repository. For example, you can query a repository for objects in the repository based on name, type (such as Catalog, Complex Join, Key, and LDAP Server), or on a combination of name and type. You can then edit or delete objects that appear in the Results list. You can also create new objects, and view parent hierarchies.
- Query a repository and view reports that show such items as all tables mapped to a logical source, all references to a particular physical column, content filters for logical sources, initialization blocks, and security and user permissions.

For example, you might want to run a report prior to making any physical changes in a database that might affect the repository. You can save the report to a file in comma-separated value (CSV) or tab-delimited format.

- When you save results, the encoding options are ANSI, Unicode, and UTF-8.

To query a repository

- 1 In the repository, right-click any object and choose Display Related > [*type of object on which you want to search*].
- 2 In the Query Repository dialog box, the type of object is prefilled.
- 3 In the Query Repository dialog box, complete the query information using [Table 24 on page 149](#) as a guide.

[Table 24 on page 149](#) contains a description of most fields and buttons in the Query Repository dialog box.

- 4 Click Query.
- 5 To save a repository query to an external file, click Save.
- 6 In the Save As dialog box, choose a type of file and an Encoding value.
- 7 To add additional columns of information to the results, click Add Columns.
- 8 In the Select information to add to the report dialog box, select the columns you want from the list and click OK.

You can re-order the columns by selecting a checked column and clicking Up or Down.

- 9 In the Save as dialog box, type a file name and select a Save as type.

10 Click Save.

Table 24. Query Repository Fields and Some Buttons

Field or Button	Description
Name	Type name to search by object name. You can use an asterisk (*) wildcard character to specify any characters. The wildcard character can represent the first or last characters in the search string. Searches are not case sensitive.
Show Qualified Name	Use this check box to display the fully qualified name of the object(s) found by the query. For example, if you query for logical tables, the default value in the Name list is the table name. However, if you select the Show Qualified Names check box, the value in the Name list changes to <i>businessmodelname.logicaltablename.columnname</i> .
Type	Select a type from the drop-down list to narrow your search to a particular type of object.
Query	Use this button when you are ready to submit your query.
Filter	Use this button to create or edit a filter for your query. After you create a filter, the filter criteria appear in the text box on the left of the button. For more information, see “Constructing a Filter for Query Results” on page 150 .
Edit	After executing a query, use this button to edit an object in the list of query results. Not all repository objects can be edited from the results list. For example privilege objects and user database sign on objects. If an object cannot be edited from the results list, the Edit button will not be available.
Delete	After executing a query, use this button to delete an object in the list of query results.
Parent	After executing a query, use this button to view the parent hierarchy of an object. If the object does not have a parent, a message appears. In the Parent Hierarchy dialog box, you can edit or delete objects. However, if you delete an object, any child objects of the selected object will also be deleted.
Mark	After executing a query, use this button to mark the selected objects. To unmark an object click the button again. You can mark objects to make them easier to visually identify as you develop metadata.
Set Icon	After executing a query, use this button to select a different icon for an object. To change the icon back to this original icon, use this button and select Remove associated icon. You can set special icons for objects to make it easier to visually identify them as having common characteristics. You may, for example, want to pick a special icon to identify columns that will be used only by a certain user group.
GoTo	After executing a query, use this button to go to the object in the Administration Tool view of the repository.

To create a new object

- 1 From the Administration Tool menu bar, choose Tools > Query Repository.
- 2 In the Query Repository dialog box, in the Type drop-down list, select the type of object you want to create.
- 3 Click New.

The dialog boxes that appear depend on the object type that you select. For more information see the section of this documentation that pertains to creating that object.

Constructing a Filter for Query Results

Use the Query Repository Filter dialog box to filter the results in the Results list of the Query Repository dialog box.

The Query Repository Filter dialog box contains five columns: an Item column and an operator/selection column to its right, a Value column and an operator/selection column to its right, and a Delete column, that lets you delete the highlighted filter.

To access the Query Repository Filter dialog box

- 1 From the Tools menu, choose Query Repository.
- 2 In the Query Repository dialog box, select an item in the Results list or select an item from the Type list, and then click Filter.

To construct a filter

- 1 From the Tools menu, choose Query Repository.
- 2 In the Query Repository dialog box, select an item in the Results list or select an item from the Type list, and then click Filter.
- 3 In the Query Repository Filter dialog box, click the Item field.
The Item drop-down list contains the items by which you can filter.
- 4 In the Item drop-down list, select the filter that you want to apply to the Results or Type object you selected in [Step 2 on page 150](#).

Depending on what you select in the Item drop-down list, other options may become available.

To construct a filter to view all databases referenced in a business model

- 1 From the Tools menu, choose Query Repository.
- 2 In the Query Repository dialog box, select Database from the Type drop-down list, and then click Filter.
- 3 In the Query Repository Filter dialog box, click the Item field.

The Item drop-down list contains the items by which you can filter.

- 4 In the Item drop-down list, select Related to.
The equals sign (=) appears in the column to the right of the Item field.
- 5 Click the ellipsis button to the right of the Value field, and in the drop-down list, choose Select object.
- 6 In the Select dialog box, select the business model by which you want to filter, and then click Select.
Your selection appears in the Value field.
- 7 Click OK to return to the Query Repository dialog box.
The filter appears in the Filter text box of the Query Repository dialog box.

To construct a filter to view all Presentation layer columns mapped to a logical column

- 1 From the Tools menu, choose Query Repository.
- 2 In the Query Repository dialog box, from the Type drop-down list, select Presentation Column and then click Filter.
- 3 In the Query Repository Filter dialog box, click the Item field.
The Item drop-down list contains the items by which you can filter.
- 4 In the Item drop-down list, select Column.
The equals sign (=) appears in the column to the right of the Item field.
- 5 Click the ellipsis button to the right of the Value field, and in the drop-down list, choose Select object.
- 6 In the Select dialog box, select the column by which you want to filter and click Select.
Your selection appears in the Value field.
- 7 Click OK to return to the Query Repository dialog box.
The filter appears in the Filter text box of the Query Repository dialog box.

You can construct more than one filter; when you do, the Operator field becomes active. When the Operator field is active, you can set AND and OR conditions.

TIP: If you are constructing a complex filter, you may want to click OK after adding each constraint to verify that the filter construction is valid for each constraint.

Comparing Repositories

This section explains how to use the Compare Repositories feature of the Administration Tool. It allows you to compare the contents of two repositories, including all objects in the Physical, Business Model and Mapping, and Presentation layers.

If you are using the Siebel Business Analytics applications repository (SiebelAnalytics.rpd) and have customized its content, you can use this feature to compare your customized repository to a new version of the repository received with Siebel Business Analytics.

For more information about merging the contents of your customized Siebel Business Analytics applications repository with that of a new version of the repository, see [“Merging Analytics Repositories” on page 153](#).

To compare two repositories

- 1 In the Administration Tool, open a repository in offline mode.
The repository that you open in this step is referred to as the *current repository*. For instructions on opening a repository, see [“Online and Offline Repository Modes” on page 29](#).
- 2 Select File > Compare from the toolbar.
- 3 In the Select Original Repository dialog box, select the repository you want to compare to the open repository.
- 4 In the Open Offline dialog box, type the password and click OK.
- 5 Use the Compare repositories dialog box to review the differences between the two repositories.

The following list contains the values in the Change column and a description:

Change	Description
Created	Object was created in the current repository and does not exist in the original repository.
Deleted	Object exists in the original repository but has been deleted from the current repository.
Modified	Object exists in the original repository but has been modified in the current repository.

The following is a list of some of the buttons in the Compare repositories dialog box and a description of the functionality provided:

Button	Functionality
Diff	Differences between the current repository and the original repository.
Edit 2	Opens created objects for editing.
Equalize	This equalizes the upgrade id of the objects. If objects have the same upgrade id, they are considered to be the same object. Not available when merging repositories.

Button	Functionality
Filter	Opens the Comparison Filter dialog box to allow you to filter the objects that appear in the Compare repositories dialog box by type of change and type of object. You can specify what you want to appear and what you want to be hidden. If you select the check box (Group created and deleted objects), the tool will filter out the child objects of created and deleted objects, allowing you to see only the parent objects. By default, all items are shown.
Find	Search by an object Name and Type (such as Initialization Block).
Find Again	Search again for the most recent Find value.
Mark	Marks the object you select. Boxes appear around created and modified objects. To remove marks, from the File menu, choose Turn off Compare Mode. Not available when merging repositories.
Save	Saves a list of the differences between the two repositories.
Select	Allows you to select a repository to compare with the current repository. Not available when merging repositories.
Stats	Provides the number of changes by Change type. During a multiuser development merge, this allows you to see an overview of the merge decisions that will take place.
View 1	Opens deleted objects in read-only mode.

Turn Off Compare Mode

This feature allows you to remove marks applied to objects while using the Compare Repositories and Merge Repositories features. The Turn off Compare Mode option is only available after you have clicked Mark during the File > Compare action. If no repository object is marked, this option is not available.

To enable the Turn Off Compare Mode

- From the Administration Tool toolbar, select File > Turn Off Compare Mode.

Merging Analytics Repositories

This section is for organizations that use the Siebel Business Analytics applications repository (SiebelAnalytics.rpd). The Merge Repository feature can also be used by customers to upgrade their custom repositories.

The merge process involves three versions of an analytics repository. The terms used in the following descriptions are the terms used in the Administration Tool user interface.

- **Original repository.** The repository you received with the previous version of Siebel Business Analytics. This section uses SiebelAnalytics.Original.rpd as an example.
- **Modified repository.** The repository that contains the customizations you made to the original repository. This section uses SiebelAnalytics.Modified.rpd as an example.

- **Current repository.** The repository that is installed with this version and is currently opened as the main repository. This section uses SiebelAnalytics.Current.rpd as an example.

During the merge process, you can compare the original repository with the modified repository and the original repository with the current repository. The Merge Repository feature allows you to decide on an object-by-object basis if you want to merge your customizations with the current repository.

Figure 14 on page 154 shows the following parts of the Merge repositories dialog box. Table 25 on page 155 contains descriptions of columns and buttons on the Merge repositories dialog box.

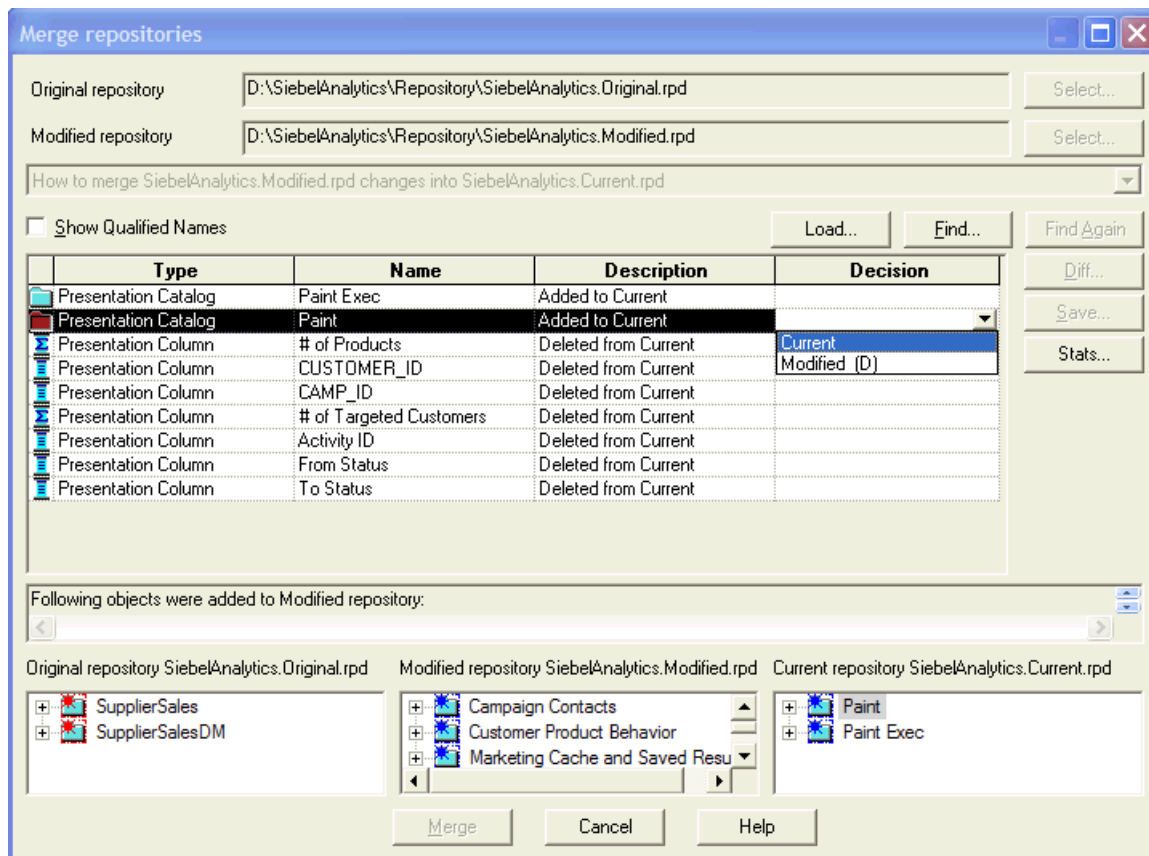


Figure 14. Merge Repositories Dialog Box

- The Original repository and Modified repository fields appear at the top of the dialog box. The Select buttons at the right of the fields allow you to select repositories.
- The read-only text box describes the type of merge that you selected.

- The decision table in the middle of the dialog box dynamically changes based on the choices you make in this window. For example, if you select the option to merge repository contents, the decision table displays the following information:

Table 25. Merge Repositories Decision Table

Column Name	Description
Type	Object type
Name	Object name
Description	Description of the changes between the original repository and the modified repository, and between the original repository and the current repository.
Decision	<p>Allows you to select an action you want to perform on the selected repository change. For examples of some results that will occur, see “Examples of the Results of Some Decision Choices” on page 155.</p> <ul style="list-style-type: none"> ■ Current. This type has no suffix. Selecting this type means that you want to leave the object in the current repository as is). ■ Modified. This type can have an A (add), a D (delete), or an AR (after renaming) suffix. ■ Mix. This type has options that apply to specific properties.
Diff (button)	Shows which properties of an object have been modified. Available for objects that are labelled Mix (not added or deleted).
Find (button)	Search by an object Name and Type (such as Initialization Block.
Find Again (button)	Search again for the most recent Find value.
Load (button)	Loads a saved decisions file from the Repository subdirectory so that you can continue processing a repository merge.
Save (button)	Saves a file containing interim changes in the Repository subdirectory so that you can stop work on the merge and continue it later. After saving the changes (decisions) you need to close the Merge repositories dialog box by clicking Cancel.
Stats (button)	In multiuser development, this allows you to see an overview of the merge decisions that will take place.

- When you select an object in the decision table, the read-only text box below the decision table describes what changes were made to that object in the current repository.

Examples of the Results of Some Decision Choices

The following examples show the results of some decision choices when the current and modified repositories are different:

- If the Description column for an object contains Added to Current, the following are choices in the Decision column and their results:
 - Selecting Current keeps the addition in the current repository.
 - Selecting Modified (D) deletes the added object from the current repository.
- If the Description column for an object contains Deleted from Modified, the following are choices in the Decision column and their results:
 - Selecting Current keeps the repository as is without deleting the object.
 - Selecting Modified (D) deletes the object from the current repository.
- If the Description column for an object contains Deleted from Current, the following are choices in the Decision column and their results:
 - Selecting Current keeps the repository as is without adding the object back into the current repository.
 - Selecting Modified (A) adds the object back into the current repository.
- If the Description column for an object contains Added to Modified, the following are choices in the Decision column and their results:
 - Selecting Current keeps the repository as is without adding the object back into the current repository.
 - Modified (A). Selecting Modified (A) adds the object back into the current repository.

NOTE: If a decision choice is Mix, the choices apply to each property level on an object. Click the ellipses button to open the Properties dialog box.

The procedure in this section explains how to use the Merge Repository feature of the Administration Tool to merge your repository customizations from a prior release with a new version of the analytics repository.

To merge versions of the analytics repository

- 1 In the Administration Tool, open the newly installed analytics repository (for example SiebelAnalytics.Current.rpd) in offline mode.
- 2 From the Administration Tool menu bar, choose File > Merge.
- 3 In the Select Original Repository dialog box, select the repository received with your previous version of the software (for example, SiebelAnalytics.Original.rpd).
- 4 Type the password and click OK.
- 5 Click Select for the Modified Repository field.
- 6 In the Select Modified Repository dialog box, select the repository that contains the customizations you made to the previous version of the repository (for example, SiebelAnalytics.Modified.rpd).
- 7 Click Open, type the password, and then click OK.
- 8 In the Decision drop-down list, select the action you want to take regarding the repository change, or accept the default action.

- 9 To locate subsequent rows with empty Decision fields, click the Decision header cell.
When all rows have a value in the Decision field, the Merge button is enabled.
- 10 Click Merge.
A message appears to let you know that the merge is complete.
- 11 From the File menu, choose Save As, and save the current repository under a new name.

Exporting Analytics Metadata to IBM DB2 Cube Views

You can convert Siebel Business Analytics proprietary metadata to an XML file and import the metadata into your DB2 database. For more information, see [“Using IBM DB2 Cube Views with Siebel Business Analytics” on page 282](#).

NOTE: The term IBM DB2 Cube Views is a registered trademark of IBM.

Creating an Analytics Multiuser Development Environment

Siebel Business Analytics allows multiple developers to work on repository objects from the same repository during group development of Siebel Business Analytics applications.

For example, after completing an implementation of data warehousing at a company, the administrator might want to deploy Siebel Business Analytics to other functional areas of the company. In this example, multiple developers need to work concurrently on subsets of the metadata and merge these subsets back into a master repository without their work conflicting with other developers.

In other organizations, a single developer might manage all development. For simplicity and performance, this developer might want to use an analytics multiuser development environment to maintain metadata code in smaller chunks instead of in a large repository.

In both situations, this is accomplished by creating projects in the repository file in the Administration Tool, and then copying this repository file to a shared network directory. Developers can check out projects, make changes and then merge the changes into the master repository.

NOTE: To perform the tasks in this section, administrators must understand the metadata creation process.

This section contains the following topics:

- [Setting Up an Analytics Multiuser Development Environment on page 158](#)
- [Making Changes in an Analytics Multiuser Development Environment on page 160](#)

Setting Up an Analytics Multiuser Development Environment

When developers check out projects, the Administration Tool automatically copies and overwrites files in the background. Therefore, it is important for you to perform setup tasks and for the developers to perform check-out procedures carefully, paying close attention to the messages that appear.

To prepare for multiuser development, you need to perform the following tasks:

- [Creating Projects for a Multiuser Development Environment on page 158](#). In the repository, create the projects that your developers need.
- [Set Up the Shared Network Directory on page 159](#). Identify or create a shared network directory that will be dedicated to multiuser development.
- [Copy the Master Repository to the Shared Network Directory on page 160](#). After creating all projects, you copy the repository file in which you created the projects to the shared network directory where it will be used as your master repository for multiuser development.

NOTE: In this section, we use the phrase *master repository* to refer to this copy of a repository in the shared network directory.

Creating Projects for a Multiuser Development Environment

A project consists of a discretely-defined subset of the metadata. Projects can consist of Presentation layer catalogs and their associated business model logical facts, dimensions, groups, users, variables, and initialization blocks.

The Administrator creates projects in a repository and copies this repository to a shared network directory. A best practice is to create projects of manageable size based on individual logical stars in the business model. For Siebel Business Analytics projects that are just beginning, the best practice is to begin with a repository with all the necessary physical table and join definitions. In this repository, you create a logical fact table as a placeholder in the Business Model and Mapping layer and a place-holder presentation catalog as a placeholder in the Presentation layer. As you add business model and presentation catalog metadata, new projects based on individual presentation catalogs and logical facts can be created.

When creating a project, the administrator selects a presentation catalog or a subset of logical fact tables related to the selected presentation catalog, and the Administration Tool automatically adds any business model and physical layer objects that are related. An object can be part of multiple projects.

After you create projects, they become part of the metadata and are available to multiple developers who need to perform development tasks on the same master repository. When defined this way, projects typically become a consistent repository after a developer checks out the projects and saves them as a new repository file.

NOTE: Only one person at a time can create projects in a master repository.

To create a project for a multiuser development environment

- 1 In the Administration Tool menu, choose File > Open > Offline.
- 2 In the Open dialog box, select the repository that you want to make available for multiuser development, and then click OK.
- 3 In the Administration Tool menu, choose Manage > Projects.
- 4 In the Project Manager dialog box, in the right panel, right-click and then select New Project.
The left pane contains the objects that are available to be placed in a project. The right pane contains the objects that you select to be part of the project.
- 5 In the Project dialog box, type a name for the project.
- 6 Perform one of the following steps to finish creating the project:
 - In the Project dialog box, select one or more logical fact tables within the business model that are related to the presentation catalog and then click Add.
The project is defined as explicitly containing the logical fact tables and implicitly containing all logical dimension tables that are joined to the selected logical fact tables (even though they do not appear in the right pane).
 - In the Project dialog box, select a presentation catalog and then click Add.
The Administration Tool automatically adds all the logical fact tables.
- 7 To remove any fact table from the project, in the right pane, select the fact table and click Remove.
- 8 Add any Target Levels, List Catalogs, Catalogs, Groups, Users, Variables, or Initialization Blocks needed for the project.
- 9 Click OK.

After you define all projects and set up the shared network directory, you can upload the new master repository to a shared network drive that all developers can access.

Set Up the Shared Network Directory

The administrator needs to identify or create a shared network directory that all developers can access and copy the appropriate repository files to that location as described in the following list:

Create a shared network directory that will be used only for multiuser development for the master repository. This directory typically contains copies of repositories that need to be maintained by multiple developers. Developers create a pointer to this directory when they set up the Administration Tool on their machines.

CAUTION: The administrator must set up a separate, shared network directory that is dedicated to multiuser development. If not set up and used as specified, critical repository files can be unintentionally overwritten and repository data can be lost.

Copy the Master Repository to the Shared Network Directory

Make a copy of the master repository file and paste it in the directory that you have dedicated to multiuser development. Projects from this master repository will be extracted and downloaded by the developers who will make changes and then merge these changes back into the master repository.

After you copy the repository to the shared network directory, you can notify developers that the multiuser development environment is ready to use.

Making Changes in an Analytics Multiuser Development Environment

Before checking out projects, developers need to set the multiuser directory in the Administration Tool to point to the shared network directory containing the master repository. This must be the multiuser development directory created by the administrator.

During check out and check in, a copy of the master repository is temporarily copied to the developer's local repository directory (\SiebelAnalytics\Repository by default). After checking out projects and making changes in a local repository file, each developer can check in (merge) changes into the master repository.

To make changes in a multiuser development environment, perform the following tasks:

- [Setting Up a Pointer to the Multiuser Development Default Directory on page 160](#)
- [Checking Out Analytics Repository Projects on page 161](#)
- [About Changing and Testing Metadata on page 162](#)
- [About Closing a Repository Before Publishing It to the Network on page 163](#)
- [Checking In Analytics Repository Projects on page 163](#)

Setting Up a Pointer to the Multiuser Development Default Directory

Before checking out projects, each developer needs to set up their Administration Tool application to point to the multiuser development directory. The Administration Tool stores this path in a hidden Windows registry setting on the workstation of the developer and uses it during checkout and checkin.

To set up a pointer to the multiuser default directory on a developer's machine

- 1 From the Administration Tool menu, choose Tools > Options.
- 2 In the Options dialog box, click the More tab.
- 3 In the More tab, next to the Multi-user development directory field, click Browse.
- 4 In the Browse for folder dialog box, locate and select the multiuser development network directory, and then click OK.
- 5 Verify that the correct directory appears in the Multi-user development directory field.

- 6 In the Options dialog box, click OK.

Checking Out Analytics Repository Projects

After setting up a pointer to the multiuser development default directory, a developer can check out projects, change metadata, and test the metadata. The Checkout option is only available when there is a multiuser development directory defined in the More tab of the Options dialog box. For more information, see [“Creating an Analytics Multiuser Development Environment” on page 157](#).

If a developer checks out a local repository and attempts to exit the application before publishing it to the network or discarding local changes, a message appears to allow the developer to select an action. For more information, see [“About Closing a Repository Before Publishing It to the Network” on page 163](#).

During checkout, the Administration Tool performs the following tasks:

- In the developer's local \SiebelAnalytics\Repository directory, the Administration Tool makes a temporary copy of the master repository and writes an entry in the log file for the master repository file ([*master repository*].rpd.log). The entries in this log file record the time and date that the temporary repository file was created.

CAUTION: If a repository with that name already exists in this location, the developer is asked to confirm overwriting the existing repository. If the developer clicks Yes, the existing repository will be immediately overwritten in the background and after the new repository is saved, the master repository file will be automatically deleted.

- In the multiuser directory on the network, the Administration Tool writes an entry in a log file ([*master repository*].log). The entry in the log file records the time and date when a local repository is created, the names of the projects, the Windows login name of the developer, and the machine name where the local repositories are saved. The following is an example of an entry in the log file:

```
##### 12/14/2003 6:41:55 PM: Created subset repositories
original Development1.rpd and Development1.rpd based on project(s) Supplier Sales KW2
Log in Name: kwolff Computer Name: KWOLFFP4
```

- In the developer's local \SiebelAnalytics\Repository directory, the Administration Tool saves a local copy of the selected projects in a new repository such as Development1.rpd. The developer makes metadata changes in this file.

Additionally, the Administration Tool writes an entry in a log file for the new local repository ([*local repository*].rpd.log). The entries in this log file record the time and date when a local repository is created and saved.

- In the developer's local \SiebelAnalytics\Repository directory, the Administration Tool saves a second local copy of the new repository, adding Original as the prefix. An example of this local copy might be OriginalDevelopment1.rpd.
- The Administration Tool writes an entry in a log file for the copy of the local repository (Original[*local repository*].rpd.log). The entries in this log file record the time and date when the copy of the local repository is created and saved.

- After the developer saves the new repository file, check out is complete. In the developer's local \SiebelAnalytics\Repository directory, the temporary copy of the master repository is automatically deleted.

CAUTION: When the developer selects and saves the projects to a local repository file, the Administration Tool does not place a lock on the projects in the master repository on the shared network drive. Therefore, nothing physically prevents others from working on the same project. To determine if a project has been checked out, you need to look in the log file in the multiuser development directory on the shared network drive.

To check out projects

- 1 From the Administration Tool menu, choose File > Multi-User Development > Checkout.
- 2 In the Select repository to extract subset from dialog box, select the master repository and then click Open.

This master repository is copied to your local machine.

- 3 In the Extract from dialog box, type your user name and password, and then click OK.
- 4 In the Browse dialog box, select the check boxes for the projects you want to import (check out), and then click OK.
- 5 In the New Repository dialog box, type a name for the new repository and then click Save.

The Administration Tool saves the new repository to your local machine and opens this repository in offline mode. This extracted repository might not be consistent.

CAUTION: In your local directory, you now have a copy of the extract of the master repository and the new repository you just saved. Do not modify the original extract copy of the master repository. Only make changes to the new repository that contains the projects that you checked out.

About Changing and Testing Metadata

Most types of changes that can be made to standard repository files are also supported for local repository files. Developers can add new logical columns, logical tables, change table definitions, logical table sources, and so on. Developers may also work simultaneously on the same project locally. It is important to note, however, that Siebel Business Analytics assumes the individual developer understands the implications these changes might have on the master repository. For example, if a developer deletes an object in a local repository, this change will be propagated to the master repository without a warning prompt.

The following modifications should not be made in a local repository:

- Hierarchy definitions. When modified concurrently by two developers, the changes will not be merged correctly during checkin.
- Project definitions. These should only be changed by the administrator in the master repository.
- Physical Connection settings. These are intentionally not propagated and developers should not test in local environments.

After making changes to a local repository, the developer can edit the local NQSConfig.ini file, enter the name of the repository as the default repository, and test the edited metadata.

NOTE: DSNs specified in the metadata need to exist on the developer's workstation.

For more information about testing the metadata, see [“Process of Completing the Setup for a Repository File” on page 143.](#)

After the local developer makes changes, tests the changes, and saves the repository locally, the local developer checks in the changes.

About Closing a Repository Before Publishing It to the Network

If a developer checks out a local repository and attempts to exit the application before publishing it to the network or discarding local changes, the following message appears:

Closing MUD repository without publishing or discarding your local changes will prevent other developers from being able to check in their work to the master repository. What would you like to do?

Publish repository, Discard local changes, Close repository and keep lock

The following is a description of these options:

- **Publish to Network.** Publishes the merged repository to the network share as the new master, releases the lock on the master, and the event is logged. This option is available after a Merge Local Changes event occurs. This option is also available on the File > Multi-User Development submenu.
- **Discard Local Changes.** Releases the lock on the master repository and records the event in the log. This option is available after a Checkout or Merge Local Changes is performed. available on the File > Multi-User Development submenu.
- **Close repository and keep lock.** This closes the repository leaving the master repository locked.

Checking In Analytics Repository Projects

After changing and testing the metadata on a local machine, the developer needs to check in the projects and merge the metadata changes into the master repository on the shared network directory. Only one developer at a time can merge metadata from a local repository into the master repository.

About Checking-In Projects

When the check-in process begins, the following actions occur:

- The Administration Tool determines if the master repository is currently locked. If not, it locks the master repository, preventing other developers from performing a merge until the current merge is complete, and records the lock in the log file.

For other developers, the Multi-User Development > Merge Local Changes option on the File menu will be unavailable until the current check-in process has been successfully completed.

- The Administration Tool automatically copies the current version of the master repository from the shared network directory to the \\SiebelAnalytics\Repository directory on the developer's machine and updates the log files in the local and shared network directories. This is necessary because the master repository in the shared network directory might have changed after the developer checked out the projects.

About Merging Metadata

The Administration begins the merge process involving the following files:

- **Original of the local (subset) repository.** Contains the state of the projects as originally extracted. This repository name begins with Original. An example of the file name for this copy might be OriginalDevelopment2.rpd. This version is stored in the same location as the modified version of the local repository.
- **Modified local (subset) repository.** Contains the extracted projects after being modified by the developer. This version is stored in the same location as the original version of the local repository.
- **Master repository in network shared directory.** This may have been modified by other developers before this merge. (For example, Master_SiebelAnalytics.rpd.)

During the merge, the Administration Tool checks for added objects and if found, a warning message appears. The following list describes what happens during this step:

- **Warning about added objects.** When a person checks out a project, they have the ability to modify that project in any way and check it back in. Deletions and modifications are ways in which the integrity of the project is maintained. However, adding objects might introduce objects into the repository that do not belong to any project. Therefore, all project related objects are checked and if a new object is found, a warning message appears.
- **Aggregation of related objects.** In the warning message, only the parent object is reported. The Administration tool aggregates all the objects to make the message more usable. For example, if a developer added a new business model, only the business model appears in the warning message to the user, not the tables, columns, dimensions, and so on.

When the developer closes the Administration Tool, the following actions occur:

- The master repository on the shared network directory is overwritten with the master repository containing the developer's changes.
- The [*master repository*].lck file is deleted. If another developer checks out the changed project from the master repository, the developer will see the changes made by the first developer.

CAUTION: The administrator needs to add newly created metadata to the project definition in master repository for it to be visible in future extracted versions of the project. For example, if a developer checks out a project, adds a new presentation catalog, and then checks it in, the new presentation catalog will not be visible in extracted versions of the project until it is explicitly added to the project definition. For instructions, see [“Creating Projects for a Multiuser Development Environment” on page 158](#).

Tracking Changes to the Master Repository

A summary of the development activities on the master repository is in the `[master_repository].log`. This log contains a record of the following activities:

- Projects that have been checked in, checked out, and when these actions occurred
- NT login name and computer name initiating the transaction
- When locks are created and removed

Differences Between the Multiuser Merge and Standard Repository Merge Processes

The multiuser development check-in process uses the same technology as the standard repository merge process with a few important differences. For more information about the standard repository merge, see [“Merging Analytics Repositories” on page 153](#).

The following list describes the differences that occur during a multiuser development merge:

- Inserts are applied automatically. Because a subset of the master repository is being used as the original repository, most objects in the master repository appear to be new. This would result in many unnecessary prompts that the developer would have to manually approve. Therefore, inserts are applied without a prompt during a multiuser development merge.
- Conflicts that are not inserts but are resolved as a result of the automatic inserts are applied without a prompt during a multiuser development merge.
- The database and connection pool properties on the server take precedence over the same properties on the developer's machine. This precedence are applied without a prompt during a multiuser development merge.

To check in projects to the master repository

- 1 From the Administration Tool menu, choose File > Multi-User Development > Merge Local Changes.
- 2 In the Lock Information dialog box, complete the following fields:
 - a In the Full Name field, type your complete name.
 - b In the Comment field, type a description of the changes that you made.
- 3 Click OK and the following occurs:
 - The master repository file from the shared network directory is copied to the local developers machine.
 - The master repository file on the shared network directory is locked. You can see a `[master_repository].lck` file on the shared network directory (hidden file). For example, `Master_SiebelAnalytics.lck`.
- 4 In the Merge repositories dialog box, verify that the Original local repository and Modified local repository file names are correct.

In the Merge repositories dialog box, there appear to be no differences among the three repositories. However, what this means is that there are no decisions that have to be explicitly made by the developer to check-in changes.

5 To see an overview of the merge decisions that will take place, click Stats.

6 In the Results dialog box, click close.

7 Click Merge.

The changes are merged and the merged local repository file opens. In the developer's local directory, a CSV file is created that contains details of the merged changes.

8 Verify that the changes made in the modified local repository are reflected in this merged local repository.

CAUTION: The merged repository has the same name as the shared repository, but this is still a local copy.

9 After you confirm all the changes, click Save.

This saves the merged repository locally, and then, uploads this repository to the shared network directory with a 000 file extension. For example, Master_SiebelAnalytics.000.

At this point, the changes made by the developer are still not saved to the master repository in the shared network directory.

10 To commit these changes to the master repository in the shared network directory, close the Administration Tool.

11 In the Siebel Analytics Administration Tool dialog box, click Yes to release the lock on the repository.

The master repository on the shared network directory is overwritten with the master repository containing the developer's changes.

Setting Up the Repository to Work with Delivers

The Siebel Business Analytics Web Server needs to deliver alerts from Delivers to entire groups and to specified email addresses, phone numbers, and so on. Delivers uses a tool called *ibot* to deliver alerts. iBots are software-based agents driven by a schedule or events that can access, filter, and perform analysis on data based upon defined criteria. For more information about iBots, see the chapter about using Delivers in *Siebel Business Analytics User Guide*.

You need to set up the Siebel Business Analytics (SA) System subject area in the Presentation layer of the analytics repository for this to function correctly.

NOTE: For information about setting the Siebel Business Analytics Web Server parameters, see *Siebel Business Analytics Web Administration Guide*.

This section contains the following topics:

■ [About the SA System Subject Area on page 167](#)

■ [Setting Up the SA System Subject Area on page 167](#)

About the SA System Subject Area

In Siebel Business Analytics, data visibility can be defined by the groups to which a user belongs. For example, when Siebel Business Analytics applications customers log on, the GROUP system session variable can be populated and the user sees certain subject areas and columns depending on the groups that exist in the variable. Although the GROUP system session variable provides Siebel Business Analytics with the groups to which each user belongs, it does not identify the users that are in each group.

The SA System subject area addresses this issue by exposing users in a group to Delivers. It also allows contact information such as email addresses to be retrieved from a database and used as delivery devices in Delivers. This allows Siebel Business Analytics administrators to set up the Analytics Server to automatically populate delivery devices and profiles for users instead of requiring users to update their My Account screen in Delivers.

Group membership is often maintained in an external database such as the Siebel transactional database and not in the analytics repository. This information can be propagated to the Analytics Server and Siebel Business Analytics Web Server through the GROUP session variable. The SA System subject area provides group membership and external email addresses to Delivers when used in conjunction with the GROUP session variable.

Setting Up the SA System Subject Area

The Siebel Business Analytics Web Server is preconfigured to look for the SA System subject area on startup and will use it if it exists. You can import any external schema that holds group membership, users' primary email addresses, and users' SMTP devices (cell phones, handhelds, and so on), and then map it to the SA System subject area.

You can add columns to the SA System table (for example, adding a provider name) by extending the Siebel S_User table. For instructions, see the chapter about configuring tables and columns in *Configuring Siebel Business Applications*. You also need to import and map the extended table.

This section contains the following topics:

- [Guidelines for Implementing the SA System Subject Area on page 167](#)
- [Setting Up the SA System Subject Area for a Stand-Alone Implementation on page 168](#)
- [Setting Up the SA System Subject Area for an Analytics Application on page 169](#)

Guidelines for Implementing the SA System Subject Area

The name for the subject area must always be SA System. User table and column names for this subject area must exactly match the column names shown in [Table 26 on page 169](#) for mapping to be successful.

The following is a list of some general guidelines to use when implementing the SA System subject area.

- If the SA System subject area is used, every user and group must be present in the data. Siebel Business Analytics does not support group membership through a mix of internal analytics repository users and external users in the SA System subject area.

- When the SA System subject area is used, it is not recommended for the user to set up a delivery profile because there is no way for administrators to control this profile. For example, the administrator loses the ability to perform a mass update of email addresses. If a user does set up a delivery profile, the delivery profile will take precedence over what is shown in the SA System subject area.
- This feature affects what users are allowed to do in the system (authorization), not who users are (authentication). For related information about database authentication, see [“About Delivers and Database Authentication” on page 340](#).
- The SA System subject area only needs to have read-access to the Administrator account and as a result, security settings are not compromised. If group membership is seen as privileged data, you can allow only the administrator to have access to this subject area.

Setting Up the SA System Subject Area for a Stand-Alone Implementation

Siebel Business Analytics standalone customers need to create and populate the SA System subject area in the analytics repository (RPD).

Customers who install the Analytics Server in a standalone environment (without Siebel Business Analytics applications) must perform the following tasks in the order shown using the data described in [Table 26 on page 169](#):

- Create tables and columns in your data source (for example your external database). For instructions, see the chapter about configuring tables and columns in *Configuring Siebel Business Applications*.
- Create and build the subject area in the analytics repository.
 - Import schema that stores the appropriate information. For instructions, see [“Creating and Administering the Physical Layer in a Repository” on page 57](#).
 - Map the tables and columns from the Physical Layer to the Business Model and Mapping layer. For instructions, see [“Creating and Administering the Business Model and Mapping Layer in a Repository” on page 103](#).
 - Map the tables and columns from the Business Model and Mapping layer to the Presentation layer. For instructions, see [“Creating and Maintaining the Presentation Layer in a Repository” on page 135](#).

The Presentation layer metadata needs to contain the SA System folder, User table, and columns as pictured in [Figure 15 on page 169](#).

Setting Up the SA System Subject Area for an Analytics Application

For Siebel Business Analytics application customers, the SA System subject area is preconfigured in the analytics repository. [Figure 15 on page 169](#) shows the table and columns in the presentation layer of the repository.

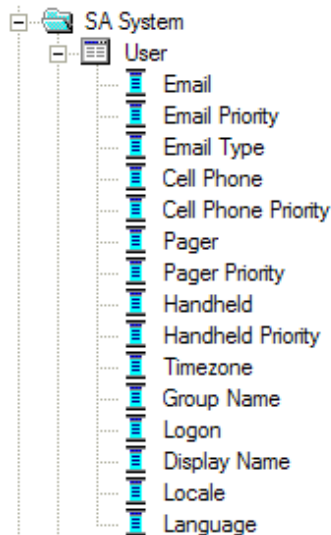


Figure 15. SA System Subject Area in the Presentation Layer

Columns in the SA System User Table

For Siebel Business Analytics application customers, the SA System subject area in the Presentation layer of the analytics repository is preconfigured for your use. [Table 26 on page 169](#) describes the columns in the User table of the SA System subject area folder. Any external schema that has the information in this table can be mapped to the SA System subject area.

Table 26. Preconfigured Columns in the SA System Area Table

Column	Data Type	Description
Logon	VARCHAR	The unique user ID of the user that will log on to the system. This cannot be null.
Display Name	VARCHAR	The full name of the user. This can be null.
Group Name	VARCHAR	The name of the group that this user belongs to. If a user belongs to multiple groups, there should be one row per group in the SA System table. This should not be null if any role-based security is based on group membership.

Table 26. Preconfigured Columns in the SA System Area Table

Column	Data Type	Description
Time Zone	VARCHAR	(This column is currently not used and exists for future use.) This should be null.
Language	VARCHAR	(This column is currently not used and exists for future use.) This should be null.
Locale	VARCHAR	(This column is currently not used and exists for future use.) This should be null.
Email	VARCHAR	The primary email address for the user. This is a complete SMTP address like joe.blow@somecompany.com. This can be null.
Email Priority	VARCHAR	This determines when an alert will be delivered to this device. The value can be any combination of the three priorities of an iBot. H for high priority, N for normal priority, or L for low priority. For example, if high, normal, and low priority alerts are to be delivered to this device, the field should be HNL. If only high and normal priority alerts are to be delivered, the field should be NL. This field should not be null if the Email column is specified. This can be null if Email is null.
Email Type	VARCHAR	This field can be one of two text strings, HTML or text. Most primary email clients can read rich MIME content (HTML with embedded images). Therefore, we recommend using HTML. However, some email clients only can read plain text email, in which case text must be used. This field should not be null if the Email column is specified. This can be null if Email is null.
Cell Phone	VARCHAR	This field is the complete SMTP address for the cell phone device that will receive text message alerts. For example, 1015551234@cellphoneprovider.com. Only text messages are sent to this device. This can be null.
Cell Phone Priority	VARCHAR	This determines when an alert will be delivered to this device. The value can be any combination of the three priorities of an iBot. H for high priority, N for normal priority, and/or L for low priority. This field should not be null if the Cell Phone column is specified. This can be null if Cell Phone is null.

Table 26. Preconfigured Columns in the SA System Area Table

Column	Data Type	Description
Pager	VARCHAR	This field is the complete SMTP address for the pager device that will receive text message alerts. For example, 1015555678@pagerprovider.com. Only text messages are sent to this device. This can be null.
Pager Priority	VARCHAR	This determines when an alert will be delivered to this device. The value can be any combination of the three priorities of an iBot. H for high priority, N for normal priority, and/or L for low priority. This field should not be null if the Pager column is specified. This can be null if Pager is null.
Handheld	VARCHAR	This field is the complete SMTP address for the handheld device that will receive text message alerts. For example, joe.blow@handheldprovider.com. Only text messages are sent to this device. This can be null.
Handheld Priority	VARCHAR	This determines when an alert will be delivered to this device. The value can be any combination of the three priorities of an iBot. H for high priority, N for normal priority, and/or L for low priority. This field should not be null if the Handheld column is specified. This can be null if Handheld is null.

Preconfigured Mapping for the SA System Subject Area

For Siebel Business Analytics application customers, [Table 27 on page 172](#) describes the preconfigured mappings for the SA system subject area. Fields that are not available in Oracle's Siebel transactional database will default to values shown in the table.

- **Overriding Defaults.** You can add user-specific values for these fields, by creating an extension table to the S_USER table. to store the user-specific defaults for these fields. Additionally, you can change any of the default values. The metadata for the following logical table can be modified to include any physical extension table.

SA User. (User)

For instructions, see the chapter about configuring tables and columns in *Configuring Siebel Business Applications*.

- **Setting Provider Info.** Typically, the cell phone and the fax numbers in Siebel Data Warehouse do not contain a provider name. Therefore, the Pager will typically be a numeric value such as 555-483-3843. To append a provider to this address, use the following guidelines:
 - If the entire company has the same provider, then you can append the provider in the column mappings.

- If users can have different providers, you need to create an extension table. For instructions, see the chapter about configuring tables and columns in *Configuring Siebel Business Applications*.

Table 27. Preconfigured Mappings for the User Table in the SA System Subject Area

Logical Column	Physical Table	Expression	Comments
Cell Phone		"	It might be mapped to S_CONTACT.CELL_PH_NUM if this field contains SMTP address.
Cell Phone Priority		"	Defaults to N
Display Name	S_CONTACT	"Real Time OLTP"."".SIEBEL.S_CONTACT_User.FST_NAME ' ' "Real Time OLTP"."".SIEBEL.S_CONTACT_User.LAST_NAME	First Name concatenated with Last Name
Email	S_CONTACT	EMAIL_ADDR	
Email Priority		'HNL'	Defaults to N
Email Type		'html'	Defaults to
Group Name	S_RESP	NAME	
Handheld		"	Defaults to an empty string
Handheld Priority		"	Defaults to an empty string
Language		'en'	Defaults to 'en'
Locale		'en'	Defaults to 'en'
Logon	S_USER	LOGIN	
Pager		"	It could be mapped to S_CONTACT.PAGER_PH_NUM if this field contains SMTP address
Pager Priority		"	Defaults to N
Time Zone	S_TIMEZONE	NAME	

8

Setting Up Mobile Analytics

This chapter contains the following topics:

- [About Mobile Analytics on page 173](#)
- [Mobile Analytics Architecture \(Server and Laptop\) on page 177](#)
- [Installing Mobile Analytics on page 179](#)
- [Process of Deploying Mobile Analytics Applications on page 179](#)
- [Creating and Testing Tables and Indexes in the SQL Anywhere Database on page 182](#)
- [Creating and Storing the Mobile Analytics Repository on page 184](#)
- [Creating the Mobile Web Catalog on page 187](#)
- [Defining Sourcing Queries for Mobile Analytics on page 187](#)
- [Creating the Mobile Application Configuration File on page 189](#)
- [Preparing Mobile Analytics Applications for Deployment on page 201](#)
- [Testing and Deploying Mobile Analytics on page 202](#)
- [Synchronizing Mobile Analytics Applications on page 206](#)
- [Synchronizing Mobile Analytics in the Background \(Silent Mode\) on page 210](#)

About Mobile Analytics

Mobile Analytics allows customers to view analytics data, Intelligence Dashboards, and queries when they cannot connect to the network to access the Siebel Business Analytics application. Typically, mobile users connect their personal machines (usually laptops) to an enterprise server running Siebel Business Analytics. After downloading a mobile Analytics application, they can disconnect their laptops from the network (for example, to travel on a business trip) and still view dashboards and queries on their laptop machines. For more information about using Mobile Analytics on a mobile machine, see *Disconnected Business Analytics Online Help*.

[Table 28 on page 174](#) contains definitions of terms used to describe Mobile Analytics.

Siebel Business Analytics provides the following mobile solutions:

- Briefing Books allow mobile users to put a static snapshot of Analytics content on their laptop to access when they are working offline. You can use the static content in Briefing Books for tasks such as managed reporting and lightweight content distribution. Briefing Books can be scheduled and delivered using iBots and is available as part of Siebel Business Analytics Web. For more information about Briefing Books, see *Siebel Business Analytics Web Administration Guide*.

- Managed Mobile Analytics is centrally administered and managed. It provides most analytical functionality that is available in the network-based Siebel Business Analytics application. After populating their local database, mobile users connect to a local dashboard through a browser and see the same UI that they would see on the Intelligence Dashboards on Siebel Business Analytics Web.

NOTE: In this guide, Mobile Analytics refers to managed Mobile Analytics. The mobile database, mobile repository, and other mobile components are interchangeably referred to as mobile, local, or mobile. When Siebel Business Analytics is used to describe components, it refers to components of the network-based Siebel Business Analytics application. For example, Analytics Server refers to the network-based Analytics Server.

Frequently Used Terms for Mobile Analytics

Table 28 on page 174 contains definitions of terminology used to explain the development and deployment of Mobile Analytics applications.

Table 28. Frequently Used Terms for Mobile Analytics

Term	Description
Data set	<p>A written definition of how to create and/or populate a specified list of SQL tables for a mobile application on a mobile machine. A data set represents an inseparable downloadable unit of data for a synchronization operation. This gives mobile users more flexibility when downloading data for a mobile application to their mobile machine.</p> <p>For example, a rapidly changing fact table may have its own data set while the relatively static dimension tables may share another data set. Advanced users might select specific data sets that they wish to download during synchronization (in the Disconnected Analytics Application Manager, on the Advanced tab), but less-experienced users might prefer the default selection of data sets.</p>
Data set family	A group of related data sets. One data set in the group, labeled the parent data set, does not depend on any other data set in the group. All other data sets in the group depend directly on the parent data set and only on the parent data set.
Disconnected Analytics Application Manager	The utility program used to synchronize, start, and delete a mobile application on a mobile machine.
Mobile application	A completely self-contained Siebel Business Analytics application that runs on a mobile machine. Application data resides in a mobile SQL Anywhere database, Analytics Server metadata resides in a mobile repository (the mobile repository), and Analytics Web metadata resides in a mobile Web Catalog.

Table 28. Frequently Used Terms for Mobile Analytics

Term	Description
Mobile application configuration file	A file in XML format that defines a particular mobile application, including associating sourcing queries and SQL Script files with data sets and identifying the location of the mobile repository and mobile Web Catalog.
Mobile machine	The machine is typically a laptop on which Siebel Business Analytics is installed with the mobile option. A mobile user connects to the enterprise server from this machine and initiates a synchronization operation to download a mobile application.
Mobile repository	The analytics repository for a mobile application downloaded from the enterprise server to a mobile machine during a synchronization operation.
Mobile Web Catalog	The Web Catalog for a mobile application downloaded from the enterprise server to a mobile machine during a synchronization operation.
Enterprise application	The fully functional Siebel Business Analytics application that runs on the enterprise server. The mobile application is a version of the enterprise application that is configured for a single user. It typically contains fewer queries (reports) and dashboards and a subset of the data.
Enterprise repository	The analytics repository for the enterprise application.
Enterprise server	The central machine that runs the complete version (as opposed to the mobile version) of Siebel Business Analytics and hosts the enterprise application. Mobile users connect their mobile machines to the enterprise server to download a mobile application.
Enterprise Web Catalog	The Siebel Business Analytics Web Catalog for the enterprise application.
Full data set	A data set that represents all data for a particular mobile application at a given point in time. Subsequent incremental updates are defined by incremental data sets. The full data set and the incremental data sets all belong to the same data set family. The full data set is the parent data set.
Incremental data set	A data set that defines additional data generated after a full data set.
Parent data set	The data set in a data set family on which all other data sets in the family depend.
Sourcing query (report)	A query that is stored in the Siebel Business Analytics Web Catalog on the enterprise server and is used to generate data files for downloading to a mobile application.

Table 28. Frequently Used Terms for Mobile Analytics

Term	Description
SQL script files	Files containing SQL commands that create and populate tables and indexes in the SQL Anywhere database on a mobile machine.
Synchronization operation	<p>The mechanism for synchronizing a mobile application on a mobile machine with the corresponding application on the enterprise server. The following are some of the tasks that this operation performs:</p> <ul style="list-style-type: none"> ■ Downloading data files generated or stored on the enterprise server to a mobile machine. ■ Loading those files into a SQL Anywhere database on a mobile machine. ■ Downloading the mobile repository. ■ Downloading the mobile Web Catalog.

Distinguishing Between the Mobile and the Enterprise Environments

The Siebel Business Analytics environment on the mobile machine (typically a laptop) is a logical equivalent to the Siebel Business Analytics environment running on an enterprise server. However, the mobile environment has been scaled down for a single user. The laptop contains its own data stored in a SQL Anywhere database. It contains a fully functional Analytics Server that accesses a mobile analytics repository, and a fully functional Siebel Business Analytics Web Server that accesses a Mobile Web Catalog. The following is a list of some of the primary differences between the mobile and enterprise environments:

- **Data.** The data in the mobile SQL Anywhere database accessed by the mobile application ultimately comes from the enterprise application (through sourcing queries). However, its size is usually reduced using filtering and aggregation.
- **Repository.** Typically, the mobile repository is much smaller than the enterprise repository, but it can contain new metadata specially designed for the mobile application. Therefore, the mobile repository is not a strictly-defined, proper subset of the enterprise repository.
- **Web Catalog.** Similarly, the mobile Web Catalog is typically much smaller than the enterprise Web Catalog. However, it too can contain local customizations such as new dashboards and queries (reports) specifically for the mobile application.

NOTE: Sourcing queries (reports) are not the same as standard reports that are displayed on intelligence dashboards. The purpose of the sourcing query is only to populate data on a mobile machine.

SQL Scripts

To create SQL Anywhere tables, and possibly indexes for those tables, on the mobile machine, administrators must create scripts with appropriate SQL commands. Each table must have a distinct corresponding script file, and all indexes for a given table share a distinct script file. For more information about SQL script files, see [“Creating and Testing Tables and Indexes in the SQL Anywhere Database” on page 182.](#)

Sourcing Queries

Sourcing queries generate the data that is stored in the SQL Anywhere database on the mobile machine. Administrators create sourcing queries using Siebel Business Analytics Web and store them in the Siebel Business Analytics Web Catalog on the enterprise server. Sourcing queries do not differ substantively from any other queries (requests) stored in the Web Catalog. Only their use for populating the SQL Anywhere database on the mobile machine distinguishes them from other queries.

Each sourcing query corresponds to a specific table in the SQL Anywhere database on the mobile machine. Therefore, the order, number, and type of columns in the sourcing query must exactly match the order, number, and type of columns in the associated table. For more information about sourcing queries, see [“Defining Sourcing Queries for Mobile Analytics” on page 187](#).

The Disconnected Analytics Application Manager

The Disconnected Analytics Application Manager runs on the mobile machine and coordinates the download process in conjunction with Siebel Business Analytics Web running on the enterprise server. The Disconnected Analytics Application Manager is installed on the laptop when you install Mobile Analytics. It allows a mobile user to download tables and application data to a laptop database. It does not upload data from the laptop to the enterprise server. For instructions about how to use the Disconnected Analytics Application Manager, see *Disconnected Business Analytics Online Help*.

CAUTION: You should not edit a DAD or SDC file. The Disconnected Analytics Application Manager is associated with the DAD and SDC file extensions.

Mobile Analytics Architecture (Server and Laptop)

[Figure 16 on page 178](#) illustrates the general architecture of Mobile Analytics. You will use two different analytics platforms, each with a set of components. One platform is on the enterprise server on your network and one is on the mobile machine. The difference between the enterprise server components and mobile server components is that the mobile components are configured for a single user.

The following list describes the types of graphics in [Figure 16 on page 178](#):

- Cylinders represent files stored on disk.
- Rectangular text boxes represent processes related to Siebel Business Analytics.
- Solid single lines with arrows represent the normal flow of data for an Siebel Business Analytics configuration (on the mobile machine or the enterprise server).

- Dashed single lines with arrows represent the flow of data during a synchronization operation from the enterprise server to the mobile machine.

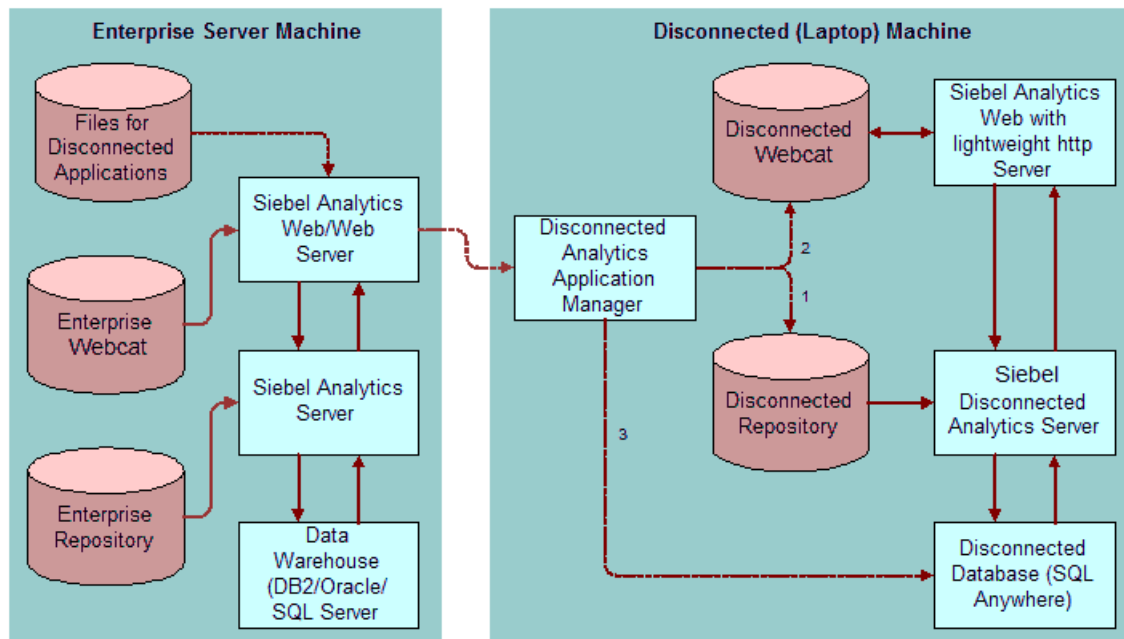


Figure 16. Mobile Analytics Architecture

Mobile Analytics Scenario

Typically, a Mobile Analytics user logs in to Siebel Business Analytics Web on the enterprise server, opens the mobile page, and chooses a mobile application to download. The Disconnected Analytics Application Manager starts on the mobile machine and begins coordinating with Siebel Business Analytics Web (on the enterprise server) to download the mobile application. [Figure 16 on page 178](#) illustrates this scenario.

The cylinder labeled Files for Mobile Applications represents the location on the enterprise server for the mobile repository, the mobile Web Catalog, and the SQL scripts associated with the mobile application. The numbers of the dashed lines in [Figure 16 on page 178](#) correspond to the numbers in the following list:

- 1 Siebel Business Analytics Web on the enterprise server sends a copy of the mobile repository to the Disconnected Analytics Application Manager that stores the mobile repository at the appropriate location on the mobile machine. The Disconnected Analytics Application Manager also changes the Analytics Server's configuration file (NQSCONFIG.INI) on the mobile machine to reference the mobile repository.
- 2 Siebel Business Analytics Web on the enterprise server sends a copy of the mobile Web Catalog to the Disconnected Analytics Application Manager. Shared content from the Web Catalog on the enterprise server is merged with the mobile Web Catalog and is stored in the mobile Web Catalog at the appropriate location on the mobile machine.

- 3 Siebel Business Analytics Web on the enterprise server downloads the SQL Scripts and data files generated from the sourcing queries to the Disconnected Analytics Application Manager. The Disconnected Analytics Application Manager then executes the SQL scripts to create the SQL Anywhere tables and loads those tables from the data files.

If the synchronization operation completes successfully, the user can click the Disconnected Analytics Application Manager's Start button to start the mobile application on the mobile machine. Note, however, that although users can download multiple applications to their mobile machines, they can only run one application at a time. Mobile Analytics does not support many concurrently running applications. Therefore, starting a mobile application will result in shutting down any currently running mobile application.

Installing Mobile Analytics

Mobile Analytics is not installed at the same time as Siebel Business Analytics. You cannot install both products on the same machine. Installing Mobile Analytics requires the following separate installations:

- **Standard Siebel Business Analytics installation.** You must install Siebel Business Analytics on your network and prepare for the mobile installation by completing the tasks in ["Process of Deploying Mobile Analytics Applications."](#)
- **Mobile client installation.** After completing the setup tasks, you need to install the mobile client on each mobile machine that needs to access a mobile application.

If you purchased Mobile Analytics, the installer detects this in the license key during installation. When prompted to select the setup type that best suits your needs, you select Mobile Client.

For installation instructions, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

NOTE: Oracle's Siebel Pharma Mobile Analytics is a preconfigured Mobile application for Oracle Pharma Sales. Several components of Mobile Analytics have been tailored for this Pharma product. These preconfigured components are installed during the Siebel Business Analytics installation in the \Siebel AnalyticsData\Disconnected\Pharma directory. For more information, see ["Pharma Mobile Analytics Administration Reference"](#) on page 417.

Process of Deploying Mobile Analytics Applications

This section describes how administrators deploy a Mobile Analytics Application to make the mobile application available for downloading from the enterprise server.

Each Mobile Analytics application contains several components such as a repository and Web Catalog that are slightly different from their counterparts on the enterprise server. Because Mobile Analytics provides only local platform functionality and a few utilities, the administrator needs to create each mobile application by building a mobile database, a customized mobile repository, a mobile Web Catalog, sourcing queries to download data, and DDLs (data definition language). After you create an application, you can test, and then deploy it.

Use the following topics to create and deploy a mobile application:

- [About the Mobile Directory Structure on page 180](#)
- [Creating and Testing Tables and Indexes in the SQL Anywhere Database on page 182](#)
- [Creating and Storing the Mobile Analytics Repository on page 184](#)
- [Creating the Mobile Web Catalog on page 187](#)
- [Defining Sourcing Queries for Mobile Analytics on page 187](#)
- [Creating the Mobile Application Configuration File on page 189](#)
- [Preparing Mobile Analytics Applications for Deployment on page 201](#)
- [Testing and Deploying Mobile Analytics on page 202](#)

About the Mobile Directory Structure

All files associated with a mobile application must reside at specific locations within the mobile directory structure.

This chapter refers to various directories and subdirectories on the enterprise server that contain files associated with a mobile application. The following example shows the mobile directory structure:

D: \Siebel AnalyticsData	(data installation directory)
Mobile	(Mobile Analytics directory)
MyApp	(application directory)
app	(application metadata directory)
data	(application data directory)
messages	(application message directory)
MyApp.xml	(configuration file for this mobile application)
YourApp	
app	
data	
messages	
YourApp.xml	

Data Installation Directory

During installation of Siebel Business Analytics, the administrator will select a directory for storing Siebel Business Analytics data (data installation directory). For example, the administrator may choose the following as the data installation directory:

D: \Siebel AnalyticsData

Mobile Directory

Mobile applications reside in the Mobile subdirectory (mobile directory) under the data installation directory. Using the same example of the mobile directory structure, the mobile applications reside in D: \Siebel Analytics\csData\Mobile.

Mobile Application Directory

Each mobile application has an associated subdirectory (mobile application directory) in the mobile directory. Using the same example of the mobile directory structure, for an application called MyApp, all its files would reside in the subdirectory D: \Siebel Analytics\csData\Mobile\MyApp. In particular, it would contain the mobile application configuration file MyApp.xml. Similarly, for an application called YourApp, all its files would reside in the subdirectory D: \Siebel Analytics\csData\Mobile\YourApp. In particular, it would contain the mobile application configuration file YourApp.xml.

NOTE: Analytics Web examines the Mobile subdirectory to identify and list the mobile analytics applications available for the user. If an application definition file and application subdirectory are present, Analytics Web reads the definition file and lists the application. To hide an application, remove the appropriate application subdirectory from the Mobile subdirectory.

Mobile Application Metadata Directory

Each mobile application directory has a subdirectory (mobile application metadata directory) that contains the metadata files of the mobile application (repository, Web Catalog and SQL script files). By convention, the directory name of the mobile application metadata directory is app, but you can change this. Using the same example of the mobile directory structure, the mobile application metadata files for the MyApp application reside in D: \Siebel Analytics\csData\Mobile\MyApp\app.

Mobile Application Data Directory

Each mobile application directory has another subdirectory (mobile application data directory) that contains the mobile application's data files (externally generated data files and preprocessed data files). By convention, the directory name of the mobile application data is data, but you can change this. Using the same example of the mobile directory structure, the mobile application data files for the MyApp application reside in D: \Siebel Analytics\csData\Mobile\MyApp\data.

Mobile Application Message Directory

Each mobile application directory has another subdirectory (hereafter called the mobile application message directory). This subdirectory contains the message localization XML files used by the mobile Web Catalog for the application. By convention, the directory name of the mobile application message is messages, but you can change this. Using the same example of the mobile directory structure, the message localization XML files for the MyApp application reside in D: \Siebel Analytics\csData\Mobile\MyApp\messages.

Mobile Application Configuration File

For more information, see [“Creating the Mobile Application Configuration File” on page 189](#).

Creating and Testing Tables and Indexes in the SQL Anywhere Database

Before you can download data to the mobile machine, you need to set up a SQL Anywhere database with the appropriate tables.

NOTE: It is recommended that you use SQL Anywhere to test creation of the mobile tables (your DDL scripts).

You will add pointers to these scripts when you create your application configuration file. When the Disconnected Analytics Application Manager initiates synchronization, the database tables are created or overwritten.

To create and test mobile tables and indexes, use the guidelines in the following topics:

- [Creating SQL Scripts on page 182](#)
- [Scenario for Using SQL Scripts to Create Mobile Tables and Indexes on page 182](#)
- [Testing SQL Scripts That Create Mobile Tables and Indexes on page 184](#)
- [Storing SQL Scripts in the Mobile Directory on page 184](#)

Creating SQL Scripts

Administrators need to perform the following tasks so that the Disconnected Analytics Application Manager can create tables and indexes in the SQL Anywhere database on the mobile machine:

- Determine the tables and indexes needed for a mobile application.
- Create corresponding scripts, creating one script for each table. Optionally, you can also create one for all indexes of a given table.

For more information about SQL syntax for SQL Anywhere, you can download the documentation (Adaptive Server Anywhere, SQL Reference) from the Sybase Web site, or you can contact Sybase for copies of the documentation.

Scenario for Using SQL Scripts to Create Mobile Tables and Indexes

A simple retail business application might keep track only of products sold and the stores where sales occur. A mobile version of this application might consist of two dimension tables (Product and Store) and a central fact table called SalesFact. Each of these three tables would need a corresponding SQL Script file for the mobile application.

This section describes the SQL Script files for these three tables.

Product.sql

```
drop table Product;

create table Product (
    P_ID integer,
    P_Name char(30),
);
```

Store.sql

```
drop table Store;

create table Store (
    S_ID integer,
    S_Name char(30),
    S_City char(20),
    S_State char(2)
);
```

SalesFact.sql

```
drop table SalesFact;

create table SalesFact (
    F_ProductID integer,
    F_StoreID integer,
    F_Timestamp datetime,
    F_Quantity smallint,
    F_Price smallint
);
```

Optionally, mobile applications may have SQL script files for creating one or more indexes on a given table. Continuing to use the simple retail business example, this section describes a SQL script for creating an index on the P_Name column of the Product table.

ProductIdx.sql

```
drop index ProductIX;

create index ProductIX on Product (P_Name);
```

Testing SQL Scripts That Create Mobile Tables and Indexes

Siebel Business Analytics administrators can test their scripts using the Sybase Interactive SQL Utility (dbisqlc.exe). Sybase documentation provides instructions about how to use this utility. You can download the documentation from the Sybase Web site and search for Interactive SQL utility, or you can contact Sybase for copies of the documentation.

Storing SQL Scripts in the Mobile Directory

After successfully testing the scripts, Siebel Business Analytics administrators should store them in the mobile application metadata directory for the appropriate application. For the retail business application scenario, this directory might be D: \Siebel AnalyticsData\Mobile\Retail\app. For a description of mobile directories, see [“About the Mobile Directory Structure” on page 180](#).

Creating and Storing the Mobile Analytics Repository

To build a mobile repository (the analytics repository for the mobile machine) a Siebel Business Analytics administrator should use the enterprise application repository as a starting point. However, because every mobile application has a unique set of requirements, this section explains the specific repository changes necessary for all mobile repositories. For more information about creating repositories, see [“Planning and Creating a Siebel Business Analytics Repository” on page 41](#).

NOTE: It is recommended that you do not make the mobile repository complicated or large (more than 20 MB). This can affect the performance of the client environment.

For development purposes, the mobile repository should be set up and tested on a laptop that mirrors the mobile user's environment as closely as possible.

The following topics explain repository changes needed for mobile repositories:

- [Assign the Mobile Database Type and Connection Pool on page 184](#)
- [Testing and Storing the Mobile Repository on page 186](#)

Assign the Mobile Database Type and Connection Pool

The following list explains the unique properties for the mobile database and mobile connection pool:

- In the physical layer of every mobile repository, the database type must be set to SQL Anywhere because that will always be the database on the mobile machine. [Figure 17](#) illustrates where to set the Database type to SQL Anywhere.

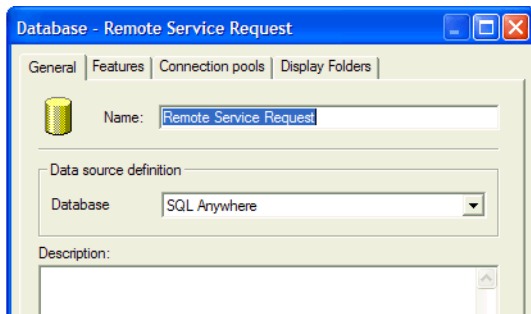


Figure 17. Mobile Data Source Definition

- [Figure 18](#) shows the connection pool parameters in the Connection Pool dialog box.

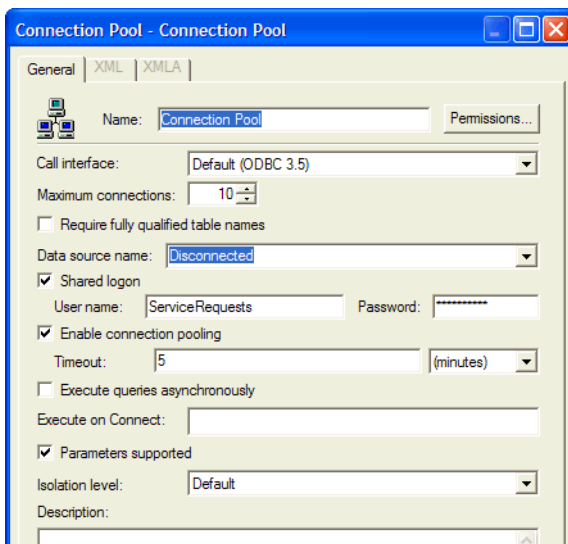


Figure 18. Mobile Connection Pool Properties

- The connection pool data source name must be Mobile because client laptops are configured during the Mobile Analytics installation to use this data source name.
- The SQL Anywhere personal database server, licensed with Mobile Analytics, cannot have more than 10 concurrent database connections. Therefore, when setting up a Mobile Analytics RPD, set the Maximum Connections value to less than or equal to 10.
- The user name and password of the connection pool must be the same as the name of the mobile application. This application name is specified in the application configuration file and the user name and password must match it. For more information, see [“Create the Application Configuration File for Mobile Analytics” on page 190](#).

- The mobile user does not need to know the user name and password for the mobile database. You can define a common user name (Administrator by default) in the repository and mobile users can use this to log in to their mobile applications.
 - For authentication purposes, you can add a standard user name in the mobile repository for all mobile users to connect to their Mobile Analytics application.
 - Data is always personalized, because it is extracted from the enterprise server using each user's name and password.
 - If your company uses Oracle's Siebel operational applications remotely, you can set up a separate connection for authentication purposes. The user name and password in this connection must be :USER and :PASSWORD. These variables allow the values to be used for authentication purposes. For more information about repository variables, see ["Using Repository Variables" on page 297](#).
- Column names in a database table cannot contain special characters such as #. Change fact table column names to contain alphanumeric characters only.

Testing and Storing the Mobile Repository

Test the mobile repository, and then store it in the mobile application metadata directory for the appropriate application. For the retail business scenario in ["Creating and Testing Tables and Indexes in the SQL Anywhere Database" on page 182](#) this would be D:\Siebel AnalyticsData\Mobile\Retail\app. For a description of mobile directories, see ["About the Mobile Directory Structure" on page 180](#).

The following is a list of high-level steps that you might use to help you test your mobile repository:

- Identify a machine to test the mobile repository. This machine can either be a network server or a laptop.
- Install the mobile client on this machine. The installation creates a mobile version of the Analytics Server that is scaled for single use.
- Create the mobile database on this machine by using dbisqlc.exe to manually create the tables.
- Deploy the mobile repository on the enterprise server by editing the nqsconfig.ini file to point to the mobile repository. For more information about the NQSConfig.INI file parameters, see *Siebel Business Analytics Platform Installation and Configuration Guide*.
- Use the mobile client to connect to the server (the default DSN name is Analytics Web) and view the schema.

CAUTION: Do not try to use Siebel Business Analytics Web yet.
- Run a sample logical SQL report based on the Presentation layer. No data will be returned because you have not populated the tables. However, this task allows you to test the integrity of your repository.

Creating the Mobile Web Catalog

To create the mobile Web Catalog (a Web Catalog for the mobile machine), you need to know how to create the dashboards and queries (reports) stored in the Web Catalog. For instructions, see *Business Analytics Web Online Help*.

To build a mobile Web Catalog, Siebel Business Analytics administrators should use the Web Catalog of the enterprise application as a starting point, deleting dashboards and reports not needed for the mobile application. Alternatively, if the mobile application needs only a few dashboards and reports, Siebel Business Analytics administrators might decide to manually create a new mobile Web Catalog.

After you create the mobile Web Catalog, store it in the mobile application metadata directory for the appropriate mobile application. For the retail business scenario in [“Creating and Testing Tables and Indexes in the SQL Anywhere Database” on page 182](#) this would be

D: \Siebel AnalyticsData\Mobile\Retail\app. For a description of mobile directories, see [“About the Mobile Directory Structure” on page 180](#).

Defining Sourcing Queries for Mobile Analytics

The sourcing query is an analytics request, created in Answers and stored in the Siebel Business Analytics Web Catalog on the network. Your sourcing queries design is linked to the mobile database table schema and should be based on how your business is organized.

A unique identifier differentiates sourcing queries from the other queries stored in the Web Catalog. You create the sourcing queries in Answers (Siebel Business Analytics Web) on the enterprise server and run them to populate the SQL Anywhere database on mobile machines. For instructions about how to create requests in Answers, see *Business Analytics Web Online Help*.

[Figure 19 on page 188](#) illustrates how sourcing queries populate the mobile tables from the analytics repository. When a mobile user downloads application data, Disconnected Analytics Application Manager executes the sourcing queries on the Analytics Web Server. Siebel Business Analytics Web runs the sourcing queries and compresses the data for downloading to the laptop. The download operation and the subsequent load into the mobile SQL Anywhere database is performed by Disconnected Analytics Application Manager.

Data is downloaded to a mobile machine using sourcing queries and the Disconnected Analytics Application Manager. The administrator creates sourcing queries that download data for a specific application. Sourcing queries are created in Answers, stored in the Siebel Business Analytics Web Catalog, and executed from the application configuration file that is run by the Disconnected Analytics Application Manager.

When a mobile user requests a download (synchronization), Disconnected Analytics Application Manager sends the request to the Analytics Web Server. Siebel Business Analytics Web uses this sourcing query to the Analytics Server to obtain data and package it into a data set. Siebel Business Analytics Web returns the application data to the mobile user's mobile database and populates the mobile database tables.

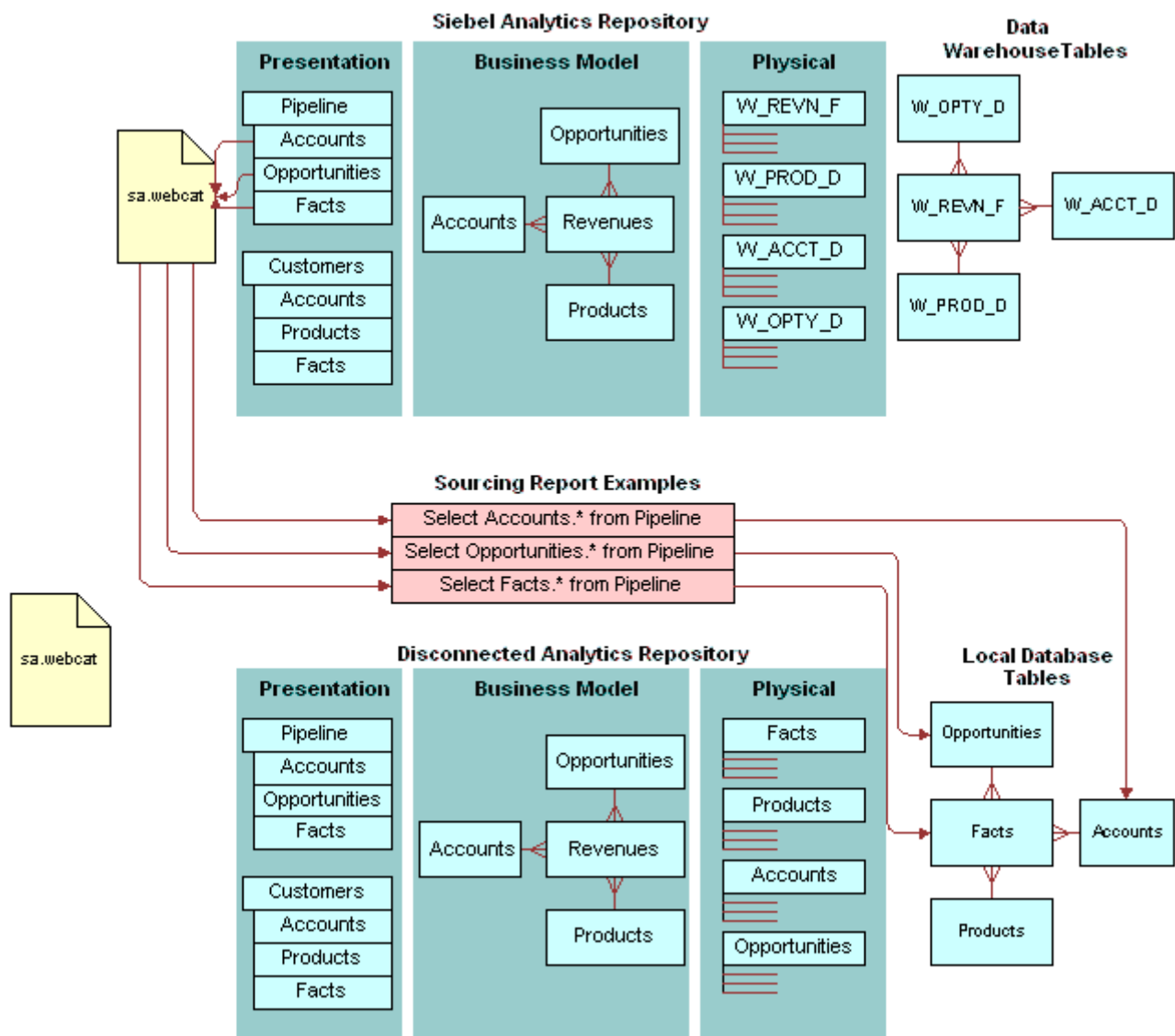


Figure 19. Mobile Analytics Sourcing Queries Populate the Mobile Tables

Each sourcing query has a corresponding table in the SQL Anywhere database. The number, type, and order of columns in the sourcing query must correspond exactly to the number, type, and order of columns in the SQL Anywhere database table to which it corresponds. For example, if the Store table for the retail database described in [“Creating and Testing Tables and Indexes in the SQL Anywhere Database” on page 182](#) contains the following columns, the sourcing query for this table needs to contain the same four columns in this exact order:

- Integer identifier
- Thirty-character name
- Twenty-character city location
- Two-character state abbreviation

NOTE: Sourcing queries can contain global filters (global filters apply to every mobile user) that the administrator adds to the source query. Additionally, mobile users can choose their own filter parameters on certain columns. To allow users to choose their filters, the administrator needs to add the appropriate columns during application configuration. For more information, see the descriptions for `<filterables>` and `<formula>` tags in [“Creating the Mobile Application Configuration File” on page 189](#).

After creating a sourcing query, Siebel Business Analytics administrators can test their sourcing queries by executing them from within Siebel Business Analytics Web. For more information, see *Business Analytics Web Online Help*.

Creating the Mobile Application Configuration File

The mobile application configuration file defines a mobile application and provides the information so that the Disconnected Analytics Application Manager and Siebel Business Analytics Web can download that application. It names all the Mobile Analytics elements required to use Mobile Analytics.

The configuration file consists of XML tags and their respective attributes. To create and maintain the mobile configuration file, you need to understand the basic concepts of XML.

Use the following guidelines when you create your mobile application configuration file:

- The base name of the file must exactly match the name of the mobile application, and the suffix of the file must be XML. In the example of the retail application described in [“Creating and Testing Tables and Indexes in the SQL Anywhere Database” on page 182](#), the mobile application configuration file would have the name Retail.xml.
- The file may contain non-ASCII characters. When the file contains non-ASCII character, the file must be saved in UTF-8 format. If you save in another format, Siebel Business Analytics Web will encounter parsing errors when reading the file.
- The file must reside in the mobile application directory. For the retail application example, the directory would be D: \Siebel AnalyticsData\Mobile\Retail. For a description of mobile directories, see [“About the Mobile Directory Structure” on page 180](#).

For a description of XML tags (and their associated attributes) that may appear in a mobile application configuration file, see [“Mobile Analytics Configuration File Reference” on page 421](#).

Create the Application Configuration File for Mobile Analytics

You create a unique application configuration file for each Mobile Analytics application. For information about where the application configuration file is stored, see [“About the Mobile Directory Structure” on page 180](#).

NOTE: If you purchased Mobile Analytics to use with your Pharma application, the application configuration file is preconfigured.

This section contains the following topics to help you create an application configuration file for your mobile applications:

- [Example of the Application Configuration File Structure \(Retail.xml\) on page 191](#)
- [Descriptions of Tags and Attributes in the Retail.xml Configuration File on page 192](#)

Example of the Application Configuration File Structure (Retail.xml)

This section contains an example for you to use when creating your application configuration files in XML format. This example uses the retail.xml file to illustrate frequently used tags and attributes. Following the example of the retail.xml file is a table containing an explanations of each tag and attribute in the order in which they appear in the file. [Table 29 on page 192](#) contains descriptions of the tags and attributes in this example.

```
<remotecfg>
  <application name="Retail"
    displayname="Retail Database Application"
    dir="app">
    <repository name="Retail.rpd"/>
    <webcatalog name="Retail.webcat"/>
    <displayname lang="en" value="Retail Database Application"/>
    <displayname lang="es" value="La Aplicación de la Base de datos al Por Menor"/>
  </application>
  <data dir="data" catalogfolder="/shared/Mobile/Retail">
    <dataset
      name="Retail"
      rank="1"
      validitytype="date-based"
      expiry="2004-12-31"
      syncdefaultfirsttime="true"
      syncdefaultsubsequenttimes="true"
      subjectarea="Retail">
      <displayname lang="en" value="Retail Data Set"/>
      <displayname lang="es" value="La Colección de Datos al Por Menor"/>
      <table name="Product">
        <sourcingreport name="Product" file="Product.csv"/>
        <tablesql name="Product.sql"/>
        <indexsql name="ProductIdx.sql"/>
      </table>
      <table name="Store">
        <sourcingreport name="Store" file="Store.csv"/>
        <tablesql name="Store.sql"/>
      </table>
      <table name="SalesFact">
        <sourcingreport name="SalesFact" file="SalesFact.csv"/>
        <tablesql name="SalesFact.sql"/>
      </table>
    </dataset>
  </data>
</remotecfg>
```

Descriptions of Tags and Attributes in the Retail.xml Configuration File

Table 29 on page 192 contains an explanation of each tag and attribute in the retail.xml file. The tags and attributes are listed in the order in which they appear in the example file. All tags and properties must be in lowercase. For a more complete list of XML tags that you might want to use in your configuration file, see [“Mobile Analytics Configuration File Reference” on page 421](#).

Table 29. Description of Tags and Attributes in the Retail.xml Configuration File

<Tag> or Attribute	Description
<remotecfg>	This tag brackets the entire file and identifies it as a mobile application configuration file.
<application>	This tag identifies the mobile application. The <application> tag can have child tags such as <repository> and <webcatalog>. Additionally, the <application> tag can have multiple <displayname> child tags. For more information, see the description of the <displayname> child tag in this table. For more information, see “Mobile Application Metadata Directory” on page 181 .
name	This attribute for the <application> tag identifies the application's internal name. This internal name is the same name as the mobile application directory (Retail in this example). The mobile application configuration file (Retail.xml in this example) resides in this directory.
displayname	This attribute for the <application> tag identifies the application's default descriptive name that appears in the Disconnected Analytics Application Manager and on Siebel Business Analytics Web's Mobile page of the enterprise server.
dir	This attribute for the <application> tag identifies the mobile application metadata directory.
<repository> (child of <application tag>)	A child tag of the <application> tag. This child tag identifies the mobile application repository that is downloaded to the mobile machine during synchronization.
<webcatalog> (child of <application tag>)	A child tag of the <application> tag. This child tag identifies the mobile application Web Catalog that is downloaded to the mobile machine during synchronization.
<displayname> (child of <application tag>)	The <application> tag can also have multiple <displayname> child tags, each of which identifies the application's descriptive name in a specified language. The descriptive name appears in the Disconnected Analytics Application Manager and on the Siebel Business Analytics Web Mobile page of the enterprise server.

Table 29. Description of Tags and Attributes in the Retail.xml Configuration File

<Tag> or Attribute	Description
lang (and) value	<p>For the <displayname> child tag, the lang attribute identifies the language, and the value attribute identifies the descriptive name. When users log in to Siebel Business Analytics Web, they specify their choice of language.</p> <p>For example, in the retail application, a user that chooses English sees a display name of Retail Database Application. A user that chooses Spanish sees La Aplicación de la Base de datos al Por Menor. A user that chooses any other language would see the default display name specified by the displayname attribute for the <application> tag.</p>
<data>	This tag identifies all data downloaded to the mobile machine as part of synchronization. For more information, see "Mobile Application Data Directory" on page 181 .
dir	This attribute for the <data> tag identifies the mobile application data directory.
catalogfolder	<p>This attribute for the <data> tag identifies the full path of the folder in the enterprise Web Catalog that contains the sourcing queries for the application. In this example, all sourcing queries for the retail application reside in the following web catalog folder:</p> <p>/shared/Mobile/Retail.</p>
<dataset>	This tag identifies a unit of synchronization. All tables specified in a data set are sent to the mobile machine as an inseparable package during synchronization.
name	This attribute for the <dataset> tag identifies the internal name of the data set.
rank	This attribute for the <dataset> tag indicates the order in which data sets will appear in the advanced area of the Disconnected Analytics Application Manager and the order in which the Disconnected Analytics Application Manager will process those data sets. The rank attribute must be a non-negative integer. The retail.xml example contains a single data set with a rank of 1.
validitytype	This attribute for the <dataset> tag identifies the type of validation performed on the data set. In this case, the Disconnected Analytics Application Manager will perform date-based validation (verifying that the current date precedes a specified expiration date). For more information, see the expiry attribute description in this table.
syncdefaultfirsttime	This attribute for the <dataset> tag indicates if the data set should be downloaded by default during the first synchronization of the data set family on a given mobile machine. For related information, see the syncdefaultsubsequenttimes attribute description in this table.

Table 29. Description of Tags and Attributes in the Retail.xml Configuration File

<Tag> or Attribute	Description
syncdefaultsubsequenttimes	The syncdefaultsubsequenttimes attribute for the <dataset> tag indicates if the data set should be downloaded by default during subsequent sync operations of the data set family on that same mobile machine. For related information, see the syncdefaultfirsttime attribute description in this table.
subjectarea	This attribute for the <dataset> tag identifies the name of the subject area in the repository of the enterprise server associated with the mobile application. Siebel Business Analytics Web on the enterprise server uses this name when storing filters for the application in the Web Catalog of the enterprise server.
<displayname> (child of <dataset tag>)	This child tag identifies the descriptive name of the data set in a specified language. The <dataset> tag can have multiple <displayname> child tags, each identifying the descriptive name of the data set in a different language. This descriptive name appears in the advanced area of the Disconnected Analytics Application Manager and on the Mobile Filter page of Siebel Business Analytics Web on the enterprise server. Users that choose a language not specified in one of the <displayname> tags will see the data set internal name (Retail in this example).
<table>	<p>This tag identifies a SQL table created on the mobile machine as part of a data set. In the retail application example, two dimension tables and a fact table were created. Each table has a separate <table> tag, and each <table> tag can have several child tags, including <sourcingreport>, <tablesql>, and <indexsql>.</p> <p>NOTE: In the <table> tag for the Product table, there is a sourcing query (report) also called Product. The output from that query goes to a file called Product.csv. During synchronization, the Disconnected Analytics Application Manager creates the Product table on the mobile machine using the script file Product.sql, and creates any associated indexes using the script file ProductIdx.sql. After creating the table and its indexes, the Disconnected Analytics Application Manager populates the table from the Product.csv file.</p>
<sourcingreport> (child of <table> tag)	This child tag identifies the sourcing query (report) used to generate the table's data.
<tablesql> (child of <table> tag)	This child tag identifies the script file used to create the SQL table on the mobile machine.
<indexsql> (child of <table> tag)	This child tag identifies the script file used to create one or more indexes for the table.

Data Set Definitions

The sample XML configuration file for the retail application in [“Create the Application Configuration File for Mobile Analytics” on page 190](#) was a simple configuration file containing a single data set with three tables. You might need only a single table in a data set or, for a more complex application, you might want to use multiple data sets.

This section discusses why administrators might want to define multiple data sets for a particular mobile application or why they might want to define a single data set with only one table.

This section contains the following topics:

- [Defining Multiple Data Sets on page 195](#)
- [Static Dimensions on page 196](#)
- [Common Dimensions on page 197](#)
- [Fact Tables and Private Dimensions on page 198](#)

Defining Multiple Data Sets

This section describes multiple data sets by using an example of a customer interaction application that tracks the following types of customer interactions:

- Service requests submitted by customers. Service requests have an owner, a status and a submission date and time.
- Activities in which customers participate such as phone conversations, on-site visits, and product demonstrations. Activities have a priority, a type and a scheduled date and time.

Figure 20 on page 196 illustrates the database schema for the application. It shows a schema with two main fact tables (Activity and SR) and their associated dimension tables. The two fact tables share the Customer and Time dimension tables.

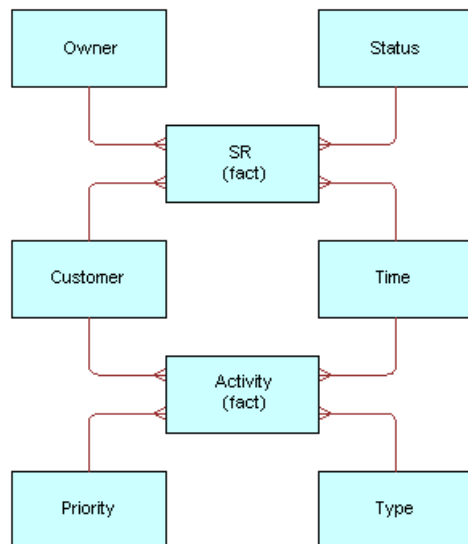


Figure 20. Database Schema for the Application with Multiple Data Sets

CAUTION: It is the administrator's responsibility to verify data consistency and integrity. For example, if you create a new data set that contains new dimensional records and do not update the corresponding facts data set, the mobile database might provide incorrect results.

Static Dimensions

Static dimensions are dimension tables that do not change for long periods of time, if at all. For example, a dimension table of United States zip (postal) codes could be a static dimension for most applications. Because static dimensions do not change often, mobile users typically need to download them only once. Therefore, administrators should put all static dimension tables for a mobile application into a separate data set.

The Customer Interaction application example in ["Defining Multiple Data Sets" on page 195](#) has one static dimension (Time). Database administrators typically preallocate many years worth of time data. As a result, the table does not change for years at a time. For example, in the Customer Interaction application, the Time dimension contains static data through the year 2010.

Example of a Data Set for a Time Dimension

The following is an example of the data set definition for the Time dimension.

```
<dataset      name="Static"
              Rank=1
              val id i tytype="date-based"
              expi ry="2010-12-31"
              syncdefaultfi rsttime="true"
              forcesync="true"
              subj ectarea="CustomerInteracti on">
  <di spl ayname l ang="en"  val ue="Data Set for Stati c Dimensi ons"/>
  <di spl ayname l ang="es"  val ue="Col ecci ón de Datos para Dimensi ones Constante"/>
  <tabl e name="Time">
    <sourci ngreport name="Ti me" fi le="Ti me. csv"/>
    <tabl esql name="Ti me. sql "/>
  </tabl e>
</dataset>
```

NOTE: A download occurs by default only during the first synchronization on a mobile machine (see the `<dataset>` attribute `syncdefaultfirsttime = "true"`). The user cannot override this attribute (`forcesync = "true"`).

Common Dimensions

Common dimensions are dimension tables that join with more than one fact table. Because common dimensions join with multiple fact tables, they must always be synchronized. Therefore, administrators should put all common dimension tables for a mobile application into a separate data set.

The Customer Interaction application example in [“Defining Multiple Data Sets” on page 195](#) contains two common dimensions (Customer and Time). Because the Time dimension already qualifies as a static dimension, we need only put the Customer table in the data set for common dimensions.

Example of a Data Set for a Common Dimension

The following is an example of the data set definition for the common Customer dimension.

```
<dataset    name="Common"
           rank="2"
           val id i tytype="date-based"
           expi ry="2004-12-31"
           syncdefaul tfi rsttime="true"
           syncdefaul tsubsequenttimes="true"
           forcesync="true"
           subjectarea="CustomerInteracti on">
  <di splayname l ang="en"  val ue="Data Set for Common Dimensi ons"/>
  <di splayname l ang="es"  val ue="Colección de Datos para Dimensi ones Comunes"/>
  <table name="Customer">
    <sourci ngreport name="Customer" fi le="Customer.csv"/>
    <tabl esql name="Customer.sql"/>
  </table>
</dataset>
```

A download occurs by default for the first synchronization (attribute syncdefaultfirsttime = "true") and all subsequent synchronizations (attribute syncdefaultsubsequenttimes = "true") on a mobile machine. The user cannot override this attribute (forcesync = "true").

Fact Tables and Private Dimensions

Private dimensions are dimensions that only join to one fact table. Because fact tables typically change independently of one another, administrators should put each fact table and its associated private dimensions into a separate data set.

The Customer Interaction application has two fact tables (Activity and SR), so each fact table and its associated private dimensions should have a separate data set.

Example of a Data Set for Fact Tables with Private Dimensions

The following is an example of the data sets for fact tables and their private dimensions.

```
<dataset    name="SR_Fact"
           rank="3"
           val i di tytype="date-based"
           expi ry="2004-12-31"
           syncdefaultfi rsttime="true"
           subj ectarea="CustomerInteracti on">
  <di splayname l ang="en" val ue="Servi ce Request Data Set"/>
  <di splayname l ang="es" val ue="Col ección de Datos de Los Pedi dos del
  Servi cio"/>
  <table name="Owner">
    <sourci ngreport name="Owner" fi le="Owner.csv"/>
    <tabl esql name="Owner.sql "/>
  </table>
  <table name="Status">
    <sourci ngreport name="Status" fi le="Status.csv"/>
    <tabl esql name="Status.sql "/>
  </table>
  <table name="SR_Fact">
    <sourci ngreport name="SR_Fact" fi le="SR_Fact.csv"/>
    <tabl esql name="SR_Fact.sql "/>
  </table>
</dataset>
<dataset    name="Acti vi ty_Fact"
           Rank="4"
           Val i di tytype="date-based"
           Expi ry="2004-12-31"
           syncdefaultfi rsttime="true"
           subj ectarea="CustomerInteracti on">
  <di splayname l ang="en" val ue="Acti vi ty Data Set"/>
  <di splayname l ang="es" val ue="Col ección de Datos de La Acti vi dad"/>
  <table name="Pri ori ty">
    <sourci ngreport name="Pri ori ty" fi le="Pri ori ty.csv"/>
    <tabl esql name="Pri ori ty.sql "/>
  </table>
  <table name="Type">
    <sourci ngreport name="Type" fi le="Type.csv"/>
    <tabl esql name="Type.sql "/>
  </table>
```

```

</table>
<table name="Acti vi ty_Fact">
    <sourci ngreport name="Acti vi ty_Fact" fi le="Acti vi ty_Fact. csv" />
    <tabl esql name="Acti vi ty_Fact. sql " />
</table>
</dataset>

```

Generating a Combined Table from Multiple Tables

You can generate a single combined table by joining a fact table to all its dimension tables.

The single table strategy results in simpler data set definitions and fewer SQL scripts to manage. It might lead to improved performance on the mobile machine because the SQL Anywhere database no longer needs to perform join operations on the combined table. However, joining the tables increases the amount of data because dimension table column values are repeated for each fact table entry.

CAUTION: Administrators should not choose the single table strategy for applications that have large fact tables and many dimension tables. It is recommended that you choose the single table strategy only if the resulting table is less than 20 MB.

For example, using the retail application example in [“Create the Application Configuration File for Mobile Analytics” on page 190](#), you can join the Product and Store dimensions to the SalesFact table producing a single table.

Example of How to Generate a Combined Table from Multiple Tables

The following example of a file called CombinedRetail.sql illustrates the SQL that would generate a combined table.

```

drop tabl e Combi nedRetai l ;
create tabl e Combi nedRetai l (
    C_ProductctID      i nteger,
    C_StoreID          i nteger,
    C_ProductName      char(30),
    C_StoreName        char(30),
    C_Ci ty            char(20),
    C_State            char(2),
    C_Ti mestamp        dateti me,
    C_Quanti ty        smal l i nt,
    C_Pri ce           smal l i nt
);

```


Example of the Simplified Retail Data Set

After combining the tables, the following example illustrates the simplified retail data set.

```
<dataset      name="SimpleRetail"
              rank="1"
              val id type="date-based"
              expiry="2004-12-31"
              syncdefaultfirsttime="true"
              syncdefaultsubsequenttimes="true"
              subjectarea="SimpleRetail">
  <displayname lang="en" value="Simplified Retail Data Set"/>
  <displayname lang="es" value="Colección Simplificada de Datos al Por Menor"/>
  <table name="CombinedRetail">
    <source report name="CombinedRetail" file="CombinedRetail.csv"/>
    <tablesql name="CombinedRetail.sql"/>
  </table>
</dataset>
```

Preparing Mobile Analytics Applications for Deployment

After you create the application configuration file, you need to test the application elements and the deployment process. After testing, you can make the application available for downloading by mobile users. You can set up a silent installation or the application can be downloaded by the mobile user using the Disconnected Analytics Application Manager. The Disconnected Analytics Application Manager is installed when the Mobile Analytics client is installed on each mobile machine.

NOTE: Disconnected Analytics Application Manager bulk loads the data into the mobile database. It does not check for duplicates and always appends data. It is the administrator's responsibility to verify data integrity.

Perform the following tasks to install a mobile application on a mobile user's laptop:

- [Creating a Silent Installation for Mobile Application Deployment on page 201](#)
- [Testing Mobile Analytics Applications on page 203](#)

Creating a Silent Installation for Mobile Application Deployment

You or mobile users can use Disconnected Analytics Application Manager to download an application on each mobile machine or you can create a silent installation. Creating a silent installation saves you time if many users will be downloading the same application. For more information about using Disconnected Analytics Application Manager, see *Disconnected Business Analytics Online Help*.

To create a silent installation, you perform an install exactly as a mobile user would perform an install, capturing all of the responses in a BAT file. It is recommended that you use a BAT file to launch the silent installation so that mobile users can map to a shared network drive, access the CD image, and execute the response (BAT) file from the command line. You can add a message to the response file that notifies mobile users that a silent installation is taking place.

The silent install does not force a restart and, most of the time, one is not needed. However, on many Windows 2000 international operating systems, a restart is required.

After building a response file you should test it on a laptop that is similar to one used by the typical mobile user for that application. For more information, see [“Testing the Installation Processes” on page 203](#).

Make sure that Mobile Analytics (client components) has been installed on the mobile machine (typically a laptop) before deploying.

To build a response file for a silent mobile application installation

- 1 Type the following command line to generate a response file:

```
setup.exe -options-record [complete path, including response file name, to shared directory]
```

- 2 Perform a complete installation, so that all responses will be captured in the specified file.

Make sure that you choose the installation directory where you want the mobile users to install the files.

- 3 To create a notification message for a silent installation, add the following lines to the BAT file:

```
@ECHO OFF
```

```
CLS
```

```
echo Siebel Analytics Mobile Client is being installed. Please wait...
```

```
setup.exe -options [complete path to response file] -silent
```

```
echo Siebel Business Analytics Mobile Client was successfully installed.
```

You test the response file when you test the silent installation.

Testing and Deploying Mobile Analytics

Before you can deploy an application to mobile users, you need to test the download process, test the silent installation (if you choose this option), and install Mobile Analytics (client components) on each mobile user's machine. For an overview of the install process, see [“Installing Mobile Analytics” on page 179](#). For more information, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

To test and install mobile applications, perform the following tasks:

- [Testing Mobile Analytics Applications](#)

- [Testing the Installation Processes on page 203](#)
- [Installing an Application Using Disconnected Analytics Application Manager on page 204](#)
- [Installing a Mobile Analytics Application Silently on page 205](#)
- [Deleting Mobile Analytics Applications on page 205](#)
- [Setting Up Mobile Machines to Operate in Silent Mode on page 205](#)

Testing Mobile Analytics Applications

Before deploying an application to mobile users, you should test each application by performing a download (with or without test data) and running the mobile application independently of the Disconnected Analytics Application Manager utility. Conducting a test in this way validates the mobile repository, Web Catalog, and tables before deployment. If you do not want to set up the database tables separately, you can test the application as a mobile user.

To test a Mobile Analytics application

- 1 Identify a machine on which you will test.
- 2 Install the Analytics Server components on this machine.
For installation instructions, see *Siebel Business Analytics Platform Installation and Configuration Guide*.
- 3 Use the mobile repository and Web Catalog on this machine, in place of the Siebel Business Analytics repository and Web Catalog.
- 4 Set up the database connections on the machine to point to your test mobile database.
If you manually load test data, you can validate your Web Catalog.
- 5 Start the Analytics Server and Siebel Business Analytics Web services.
- 6 Login to each application and run queries.
- 7 Debug when errors occur.

Testing the Installation Processes

After you test the application, you need to test the installation process. Test installing the application using Disconnected Analytics Application Manager. If you choose a silent installation, test this process.

Installing an Application Using Disconnected Analytics Application Manager

You download a mobile application using Disconnected Analytics Application Manager. This section contains general instructions. For more information about installing and using Mobile Analytics, see *Disconnected Business Analytics Online Help*.

NOTE: At this time, synchronization is a one-way process only. You can download data from the enterprise server database but you cannot upload data from the mobile repository a laptop to an enterprise server database.

To test downloading a mobile application

- 1 Identify a mobile machine on which you will test.
- 2 Install the Mobile Analytics client components on this machine.
For installation instructions, see *Siebel Business Analytics Platform Installation and Configuration Guide* and ["Installing Mobile Analytics" on page 179](#).
- 3 Using the mobile machine, Start the Disconnected Analytics Application Manager by choosing Start > Programs > Siebel Analytics 7.x > Disconnected Analytics Application Manager.
- 4 In the Disconnected Analytics Application Manager, download the application.
For detailed instructions, see *Disconnected Business Analytics Online Help*.

To test downloading data to a mobile database

- 1 Using the mobile machine, open Siebel Business Analytics on the network and click the Mobile link in the upper right corner.
- 2 The mobile applications that have been deployed appear.
- 3 Click Update Data.
- 4 In the File Download dialog box, click Open.
- 5 In the Disconnected Analytics Application Manager, you can download applications and data.

To test the application data on a mobile machine

- 1 Start application and run queries.
For more information, see *Disconnected Business Analytics Online Help*.
- 2 Debug when errors occur.

Installing a Mobile Analytics Application Silently

After you build a response file that contains all installation responses, the mobile user initiates the install by running this response file. The mobile user can temporarily map a network drive to the shared path with the CD image and the response file. From a command window, the user executes the BAT file.

NOTE: The install is completely silent unless you provide a notification message to the mobile user.

Deleting Mobile Analytics Applications

You might need to delete a mobile application from the enterprise server if you do not want anyone to download that application.

NOTE: For instructions about deleting mobile applications from a laptop, see *Disconnected Business Analytics Online Help*.

To delete a mobile application

- 1 Delete the application directory and files from the \Siebel AnalyticsData\Mobile\ directory.
- 2 Login to Siebel Business Analytics and navigate to Answers.
- 3 In Answers, delete sourcing queries from the Siebel Business Analytics Web Catalog.

For more information about Answers, see *Siebel Business Analytics Web Administration Guide*.

Setting Up Mobile Machines to Operate in Silent Mode

If you have administrator security, you can run Mobile Analytics in silent mode or set up Mobile Analytics to start automatically when you turn on or reboot a mobile machine. When the Siebel Business Analytics icon appears in the system tray at the bottom right of the taskbar, Mobile Analytics is running. The following describes how to run Mobile Analytics in the background (silent Mode):

- **Automatically start in silent mode.** To set up a mobile machine to automatically operate in silent mode, the administrator can insert the following command in the startup folder of the mobile machine:

```
sadi s.exe /s
```

Mobile Analytics will be loaded and available for querying when you start up the mobile machine. For more information about adding commands to the startup folder, see your Microsoft Windows manual.

- **Manually start in silent mode.** The administrator can open a command line window and type the following command:

```
sadi s.exe /s.
```

Synchronizing Mobile Analytics Applications

Synchronization is the mechanism that downloads data from the enterprise analytics machine to the mobile machine for each user. The first time synchronization occurs, all historical data is downloaded. After the first synchronization, the incremental synchronization feature allows incremental data downloads.

The first time users synchronize their application, they will be asked to create a username and password. This is controlled by the SETDBUSER parameter default value = YES in the Windows registry. After a user types their username and password, the value changes to NO. As a result, subsequent synchronizations do not require a username and password.

Currently, synchronization only downloads data to the mobile machine. It cannot upload data from the mobile machine to the enterprise machine.

A user can download data in the following modes:

- **Online mode.** The user submits the data set generation query in real time. The server generates and returns the data to the user. In this mode, the user must wait until the entire data set has been downloaded to their machine. Therefore, online synchronization should only be used occasionally and when a specific need requires it.
- **Preprocessed mode.** The administrator schedules the data set generation on the server. This requires that you have installed and set up Delivers. The data sets are generated for each user and maintained until a new data set is generated. The Analytics Server does not delete the data sets. When a user synchronizes their application, the preprocessed data sets for that user are identified and then downloaded by the Disconnected Analytics Application Manager. Therefore, the amount of time required for synchronization depends on the speed of this compressed data set download.

This section discusses the following topics about data synchronization:

- [About Preprocessed Synchronization on page 206](#)
- [About Incremental Synchronization on page 208](#)
- [Synchronizing Mobile Analytics in the Background \(Silent Mode\) on page 210](#)

About Preprocessed Synchronization

Mobile allows the administrator to balance the load on their servers by preprocessing sourcing queries before users synchronize their data. To set up preprocessed synchronization, the administrator schedules the sourcing queries for a mobile application to run during off-peak hours.

NOTE: It is recommended that you set up your applications to be synchronized in preprocessed mode. This will minimize the server loads during normal business hours and allow the server to create mobile data sets during non-peak usage hours.

Preprocessed data sets do not require configuration for the Disconnected Analytics Application Manager. Typically, mobile users are not aware that they receive preprocessed data. In Preprocessed mode, the Disconnected Analytics Application Manager can detect whether the data set on the local machine is up to date. A data set in preprocessed mode is considered out of date if the date of the most recent synchronization of a data set is earlier than the last modified times for any report in that data set.

This section provides guidelines for setting up a preprocessed application.

NOTE: Before following these guidelines, you need to be familiar with how to set up Mobile Analytics applications and Oracle's Siebel Scheduler. For more information about Siebel Scheduler, see *Siebel Business Analytics Scheduler Guide*.

Setting Up Syncmode in the Application Configuration File

Syncmode is the attribute in the application file that allows a data set to run in preprocessed mode. When the Syncmode attribute is set to preprocessed and a mobile user initiates synchronization, Siebel Business Analytics Web on the enterprise server looks for existing data files to download and only runs the sourcing query if no preprocessed files exist. For more information about Syncmode, see [Appendix C, "Mobile Analytics Configuration File Reference."](#)

After you modify the syncmode attribute in the configuration file, store the sourcing queries in the data directory for the application so that they will be available to users when they log on. For more information about where to store the sourcing queries, see ["Mobile Application Data Directory" on page 181](#).

Mobile Server Configuration

The administrator needs to use the guidelines in this section to schedule the execution of preprocessed sourcing queries. For more information about iBots, see the chapter about using Delivers in *Siebel Business Analytics User Guide*.

In Delivers, create a new iBot and define it using the following guidelines.

- In the Delivery Content tab, specify the sourcing queries to be run in preprocessed mode by clicking Select Content.

The content should be a dataset from your mobile application. In the tree on the left of the window, expand one of the mobile applications, and then select the dataset.
- In the Schedule tab, specify a schedule for these queries. If you choose to start immediately, the iBot runs as soon as you save it.
- In the Recipients tab, specify the users for which these queries need to be run.

NOTE: If you set up a group as a recipient, the sourcing query runs once for each user at the scheduled time.
- In the Destinations tab, select the Mobile Application Cache check box and clear the check boxes for all other options.
- Save this iBot. It will be saved in your My iBots folder in the Web Catalog on the enterprise server.

About Incremental Synchronization

Typically, the first data downloaded by mobile users from an enterprise environment is a full download of historical data. After the full download, mobile users periodically download updates of data. This requires that the Analytics Server allows the mobile user to download most recent data on a periodic basis.

Incremental synchronization allows incremental data downloads. Because different organizations have different requirements, administrators can set up incremental data in various ways, depending on their user characteristics and organizational policies. Administrators can control which users download incremental data sets and how often. They can also control what each data set contains.

Incremental Data Sets

The primary difference between incremental synchronization and a full download is the way it handles data integrity and data set validity. In a typical environment, each mobile analytics application contains at least one full data set and multiple incremental data sets. A full data set is a data set that does not depend on any other data set. Full data sets create tables and then load the data for the first time. For incremental data sets, the Disconnected Analytics Application Manager appends data using the SQL scripts specified in the incremental data set. The administrator needs to set up the SQL scripts correctly for this to work.

Incremental data sets are defined in the same way as full data sets, by defining a data set in a configuration file. While an application can have multiple data sets, each data set has its own configuration file. A few special tags differentiate incremental data sets from a full data set. These tags also drive the behavior of the Disconnected Analytics Application Manager, when it downloads the data set definition. While the tags are defined in [Appendix C, "Mobile Analytics Configuration File Reference"](#) and *Siebel Business Analytics Platform Installation and Configuration Guide*, the general purpose for using these properties is explained in the following paragraphs.

An incremental data set assumes that the table has been created and that the base data has been loaded. Different users may have downloaded data on different dates, so their data may differ depending on the following factors:

- Last update date. The last date that the user synchronized.
- Today's date. The date on the user's mobile machine.
- Data period characteristics. The start and end date period in which the data was created. For example, the data could be one week of data.

The Disconnected Analytics Application Manager does not automatically manage data integrity when the Disconnected Analytics Application Manager loads data from multiple data sets into the tables. Administrators should make sure that the data loaded does not create duplicates.

Data period characteristics are distinct from the validity checks and the dates that drive the validity checks. The data period or ranges are decided by the sourcing queries and how often the enterprise data itself is updated. For example, dimensional data is usually less dynamic than facts. Transactional data updates are tracked by the database itself as well as fields such as Last Updated Date. A sourcing query can pick up changed data fields by querying for the last updated date. Alternatively, a sourcing query can pick up data using the Created Date field. The Disconnected Analytics Application Manager does not use these dates to validate data. It is the responsibility of the Administrator to make sure that the range of dates are coordinated with the synchronization requirements and frequency of users.

Incremental Data Set Validity

To make sure that a mobile user has downloaded a full data set before they load incremental data, the Disconnected Analytics Application Manager performs certain validity checks. Validity checks make sure that data sets are loaded in a particular order and helps administrators maintain data integrity in the mobile user's mobile database.

This section contains descriptions of the types of validity checks.

Date-Based Period Validity

Date-based periods are based on calendar dates and allow administrators to use key dates to control the use of data sets.

- **Start Date.** This tag is the date on which the data set becomes valid. For example, Last Update might represent the date on which the user synchronized for the last time. To determine if a data set is valid, the start date is compared with the value of Last Update. The table is valid if the Last Update date of the table is before the Start Date of a data set.
- **Expiry Date.** An expiration date for a data set. This date is always compared to the date of the current data on the mobile machine.

Expiry dates are used primarily to make sure that mobile users do not continue to use certain data sets. Assigning an expiry date is similar to removing a data set from the enterprise server to prevent it from being used.

Rolling Periods

Rolling periods allow the administrator to create rolling windows of validity periods. This allows the administrator to control how users download recurrent or periodically created data sets, without having to update dates in the configuration files. A data set is valid if the following rule applies:

Last update date of table + validity period is after current date

The periods can be set in various units ranging from days, weeks, or months.

Data Set Validity Checks

Data sets are a data delivery unit that combines multiple tables. Data sets allow Administrators to organize their tables in logical groupings and control their creation as well as availability. Validity tags explained in [“Incremental Data Set Validity” on page 209](#) are applied at the data set levels. While data validity is specified at the data set level, it is enforced at the table level. This is to handle cases where the same table can exist in multiple data sets. A table can occur in the full data set as well as the incremental data set. Each table then inherits the validity conditions imposed on the data set. The Disconnected Analytics Application Manager then applies the validity conditions on each of the tables. For any data set to be valid, every single table in that data set has to be valid. Suppose a data set contains two tables. Table A is also present in one other data set, while Table B is present in two other data sets. Thus the last time the user updated table A may be different from that of table B. In order for the data integrity to be maintained, it is important that each table's validity is checked before making a data set available for synchronization.

Dependencies

A typical organization would have multiple incremental data sets, each with dependencies on other data sets. In order to manage the synchronization order, administrators can use the `<dependson>` tag. Using this tag, data sets can be configured such as users' are forced to follow certain order of synchronizations.

Data Integrity

The Disconnected Analytics Application Manager does not check for the data integrity when it loads downloaded data into the mobile tables. Administrator should note this when designing the data sets.

Synchronizing Mobile Analytics in the Background (Silent Mode)

To integrate dashboards with Oracle's Siebel Mobile Web client, you need to set up mobile analytics to run in the background (silent mode) using the following syntax.

Sync	(Required)
-u U[User Name]	(Required)
-p P[Password]	(Required)
-f F[Dad File / URL Link]	The DAD file name. Only if you know the DAD file name on the server.
-s S[Server]	Machine name. URL is constructed from machine & application name
-a A[Appl i cati on Name]	Required to be used in conjunction with server
-d D	(Optional) To just download files, not update, the files won't be written to disk.
-l L[Log Fi l ename]	(Optional)
-o O[Output Fol der]	(Optional) Primarily used for testing.

CAUTION: If you type a space between a parameter and its corresponding value, you will get an error such as Cannot Find DAD File. For example, type `-aPharma`, not `-a Pharma`.

Pharma Mobile Analytics Example

The command line interface is used extensively by Oracle's Siebel Pharma application and Pharma Mobile Analytics so that synchronization can be performed on a batch basis on weekends or nights. Pharmaceutical sales representatives typically synchronize multiple mobile analytics data sets using a single synchronization process. This process runs synchronization scripts for all their applications including Siebel operational applications, Mobile Analytics, and Outlook.

Pharma Mobile Analytics synchronization typically is run as a set of scripted commands that are embedded in a script and run in the background. Pharma Mobile Analytics users typically do not use Disconnected Analytics Application Manager synchronize.

The following is an example of the command line synchronization for Pharma Mobile Analytics:

- Type the following on the command line if the server and application name are specified:

```
sync -uuser -ppassword -sserver -aPharma where server is <hostname>: <port>
```

NOTE: Lowercase (analytics) is the default on the command line if the server and application name are specified.

- Type the following on the command line if the URL of the DAD file is specified:

```
sync -uuser -ppassword -f"http://server/Analytics/saw.dll?SyncDAD&app=Pharma"
```


9

Administration Tool Utilities and Expression Builder

This section describes the various utilities and wizards contained in the Administration Tool. It also describes the Expression Builder and provides instructions for creating constraints, aggregations, and other definitions within a repository.

This section contains the following topics:

- [Utilities and Wizards on page 213](#)
- [Expression Builder on page 219](#)

Utilities and Wizards

The Administration Tool provides a number of wizards and utilities to aid you in performing various tasks. This section provides a description of the following utilities and wizards:

- [Time Series Wizard on page 213](#)
- [Synchronize Aliases on page 214](#)
- [Replace Columns or Tables Wizard on page 214](#)
- [Siebel Business Analytics Event Tables on page 215](#)
- [Externalize Strings on page 215](#)
- [Rename Wizard on page 216](#)
- [Update Physical Layer Wizard on page 216](#)
- [Generating Documentation of Repository Mappings on page 217](#)
- [Removing Unused Physical Objects on page 218](#)
- [Extracting Analytics Metadata Using Dependency Tracking on page 218](#)

Time Series Wizard

The Time Series Wizard automates the process of creating measures and calculations to support historical time comparison analyses. It is recommended that you use the right-click menu to start this wizard because it typically takes less time to open using this method.

To start the Time Series Wizard

- 1 In the Business Model and Mapping layer, right-click a business model.
- 2 Choose Time Series Wizard.
- 3 Follow instructions in each window of the wizard.

Synchronize Aliases

Use this utility to synchronize the structure of physical layer alias objects with the physical table for which they are an alias. For example, if a column data type changes in the original table, this utility adjusts the data type in the alias.

To synchronize aliases

- 1 From the Tools menu, choose Utilities.
- 2 In the Utilities dialog box, select Synchronize Aliases, and then click Execute.
- 3 From the View By drop-down list, select whether you want to view the list of tables with aliases either by the original physical tables or by the alias tables.

If you select Original Tables, they appear in the left pane, and when you click a table, its alias appears in the right pane. The reverse is true if you select Aliases.
- 4 (Optional) Select the option Synchronize Primary Keys to compare or synchronize primary keys.

If this option is not selected, primary keys are not considered during the synchronize operation.
- 5 (Optional) Select the option Show Qualified Names to display qualified names in the Original Tables pane.

The fully qualified names are based on the physical object names in the repository. If this option is not selected, only the names of the tables are displayed.
- 6 To determine whether a physical table and its aliases are already synchronized, click the original table and its alias to select them, then click Compare.

An informational message will tell you whether the tables are synchronized.
- 7 To select all original tables and their aliases, click Select All, and then click Synchronize to synchronize all logical table and column aliases with their associated physical tables.
- 8 To synchronize an individual table, click the original table and its alias to select them, then click the Synchronize button.
- 9 Click the Close button.

Replace Columns or Tables Wizard

The Replace Wizard automates the process of replacing physical columns or tables in logical table sources by allowing the administrator to select the sources from those displayed. The wizard prompts the administrator to replace columns as well as tables.

To start the Replace Column or Table Wizard

- From the Tools menu, choose Utilities > Replace Column or Table in Logical Sources, and then click Execute.

Siebel Business Analytics Event Tables

This utility allows you to identify a table as a Siebel Business Analytics event polling table. An event polling table is a way to notify the Analytics Server that one or more physical tables have been updated. Each row that is added to an event table describes a single update event. The cache system reads rows from, or polls, the event table, extracts the physical table information from the rows, and purges cache entries that reference those physical tables. For more information about event tables, see [“Cache Event Processing with an Event Polling Table” on page 268](#).

To start the Siebel Business Analytics Event Tables utility

- From the Tools menu, choose Utilities > Siebel Analytics Event Tables
- Click Execute.

Externalize Strings

This utility is primarily for use by translators to translate Presentation layer catalogs, tables, columns, and their descriptions into other languages. You can save these text strings to an external file with ANSI, Unicode, and UTF-8 coding options.

CAUTION: When using this utility, translators should work closely with Oracle's Siebel Business Analytics contacts to verify that the correct process is followed for each situation.

To translate a string

- 1 In the Presentation layer, right-click a presentation catalog and choose Externalize Display Names.
- 2 In the Presentation layer, right-click the same presentation catalog and choose Externalize Descriptions.

In the right-click menu, both options appear with a check next to them.

- 3 From the Tools menu, choose Utilities > Externalize Strings, and then click Execute.
- 4 In the Externalize Strings dialog box, click a presentation catalog in the left pane.

NOTE: You can select all catalogs at once or select them individually and create a separate string file for each one.

In the right pane, the translated values and the original strings (names) appear. These will be placed in session variables for use by Siebel Business Analytics Web.

- 5 Click Save.
- 6 In the Save As dialog box, choose a type of file and an encoding value and click Save.
- 7 In the Externalized Strings dialog box, click Close.
- 8 (Optional) To clear the check marks next to these options, right-click the presentation catalog and click each option.

Rename Wizard

The Rename Wizard allows you to rename presentation and Business Model and Mapping layer tables and columns. It provides a convenient way to transform physical names to user-friendly names.

NOTE: Renaming the presentation layer columns will reset the Use Logical Column Name property to false. It is recommended that you rename the business model layer logical columns instead.

To start the Rename Wizard

- From the Tools menu, choose Utilities > Rename Wizard, and then click Execute.

Update Physical Layer Wizard

This wizard allows you to update database objects in the Physical layer of a repository based on their current definitions in the back-end database.

NOTE: This wizard is not available for repositories that are opened in read-only mode, because they are not available for updating.

When the wizard processes the update, the server running the Administration Tool connects to each back-end database. The objects in the Physical layer are compared with those in the back-end database. Explanatory text alerts you to differences between objects as defined in the database in the Physical layer and as defined the back-end database, such as data type-length mismatches and objects that are no longer found in the back-end database. For example, if an object exists in the database in the Physical layer of the repository but not in the back-end database, the following text is displayed:

Object does not exist in the database

NOTE: The wizard does not add columns or tables to the repository that exist in the back-end database but not in the repository. Additionally, the Wizard doesn't update column Key assignments. It checks that there is a column in the repository that matches the column in the database, and then, if the values don't match, the wizard updates the type and length of the column in the repository.

The connection pool settings for each database need to match the connection pool settings used when the objects were last imported into the Physical layer from the back-end database. For example, for Oracle, the connection pool may be set to native OCI, but an Oracle ODBC source must be used for the update. In this case, you would set the connection pool to the Oracle ODBC setting used for the import. For more information about connection pool settings, see [“Setting Up Connection Pools” on page 65](#).

To update objects in the Physical layer

- 1 From the Tools menu, choose Utilities > Update Physical Layer, and then click Execute.

The databases in the Physical layer of the repository are listed in the left pane of the wizard.

- 2 In the Update Physical Layer Wizard dialog box, select the databases that you want to update in the left pane, and then click Add.

The databases move to the right pane.

- 3 To remove a database from the update list, select it and click Remove.
- 4 After you select the objects that you want to update in the Physical layer, click Next.
- 5 In the next window, select the connection pool for each database that you want to update and then click Next.

The wizard alerts you if it needs to check any objects out.

- 6 Review the information about each update.

NOTE: You can sort the rows (toggle between ascending order and descending order) by clicking the Name column heading.

- 7 If you decide that you do not want the wizard to update a particular object in the Physical layer, click the Back button and remove the object.

- 8 Click Finish.

The wizard updates the objects in the Physical layer, and then closes automatically.

- 9 On the Administration Tool toolbar, click File > Save to save the updated objects in the Physical layer.

Generating Documentation of Repository Mappings

The Repository Documentation utility documents the mapping from the presentation columns to the corresponding logical and physical columns. The documentation also includes conditional expressions associated with the columns. The documentation can be saved in comma separated, tab delimited, or XML format.

To run the Repository Documentation utility

- 1 From the Tools menu, choose Utilities.
- 2 In the Utilities dialog box, select Repository Documentation, and then click Execute.
- 3 In the Save As dialog box, choose the directory where you want to save the file.
- 4 Type a name for the file.
- 5 Choose a type of file and an Encoding value and click Save.

Current encoding options are ANSI, Unicode, and UTF-8.

Removing Unused Physical Objects

Large repositories use more memory on the server and are harder to maintain. Additionally, development activities take longer on a large repository. This utility allows you to remove objects that you no longer need in your repository. You can remove databases, initialization blocks, physical catalogs, and variables.

To remove unused physical objects

- 1 From the Tools menu, choose Utilities > Remove Unused Physical Objects, and then click Execute.
- 2 In the Remove Unused Physical Objects dialog box, from the Type drop-down list, select the type of object.
- 3 In the list of objects, verify that only the objects that you want to remove are checked.
Below the list of objects, the number of checked and the total number of objects appears.
- 4 To remove the checked objects, click Yes.
- 5 To cancel, click No.

Extracting Analytics Metadata Using Dependency Tracking

The Dependency Tracking utility allows you to extract analytics metadata to a flat file so that it can be loaded into Excel and RDBMS. You can query the resulting file to answer questions such as "if I delete physical column X, what logical columns will be affected?" or "How many places in the business model refer to the physical table W_SRVREQ_F". Then you can establish dependency relationships among elements in the repository.

CAUTION: Excel only allows data sets of 65,536 rows. It is very easy to exceed this in a repository such as SiebelAnalytics.rpd. Therefore, you may wish to run dependency tracking on a subset of the repository by extracting out relevant business models into a new project.

Dependency Tracking creates a comma-separated values file, a tab-separated values file, or an XML file that shows the connections between the presentation and physical layers in the current repository. This file can be imported into a repository as a physical layer (for example, using Import from XML).

NOTE: This file excludes any information about repository variables.

To use dependency tracking to extract analytics metadata

- 1 From the Tools menu, choose Utilities.
- 2 In the Utilities dialog box, choose Dependency Tracking and click Execute.
- 3 In the Save As dialog box, type a file name.
- 4 From the Save as type drop-down list, select the type of file.

5 Click Save.

When executed, it shows the Save As dialog (exactly same behavior as Repository Documentation).

Expression Builder

You can use the Expression Builder dialog boxes in the Administration Tool to create constraints, aggregations, and other definitions within a repository. The expressions you create with the expression builder are similar to expressions created with SQL. Except where noted, you can use all expressions constructed with the expression builder in SQL queries against the Analytics Server.

For information about using SQL with the expression builder, see [“SQL Syntax and Semantics” on page 367](#). For information about the SQL functions supported by the Analytics Server, see [“SQL Reference” on page 376](#).

This section includes the following topics:

- [About the Expression Builder Dialog Boxes on page 219](#)
- [Expression Builder Toolbar on page 221](#)
- [Folders in the Selection Pane on page 221](#)
- [Example of Setting Up an Expression on page 223](#)
- [Navigating Within the Expression Builder on page 224](#)
- [Building an Expression on page 224](#)

About the Expression Builder Dialog Boxes

You can access the expression builder from the following dialog boxes:

- Logical Table Source—Content tab
- Logical Table Source—Column Mapping tab
- Logical Column—General tab
- Logical Column—Aggregation tab
- Logical Foreign Key
- Physical Foreign Key
- Session Variable—Variable tab
- Static Repository Variable—Variable tab

When creating expressions in the Expression Builder dialog boxes, you can search the categories pane and building blocks pane. When you type a value into the search box, it filters out the non-matching strings and only the ones that match will appear. After typing search criteria in a search box, you can move up and down the list using arrows and tab between the first search box and the second search box. To return to the full list of results, you delete the search string from the Search field.

When you locate the item you want to insert into the expression, select it and click Insert in the dialog box or press Enter on your keyboard. The item you selected will appear in the expression in the expression box.

When you first open the Expression Builder dialog box, the items are not sorted. When checked the Sort Panes check box sorts all items in the panes. As soon as you select the check box, the panes are automatically redrawn without changing the contents of the panes or your filtering criteria.

Figure 21 shows an example of an expression builder and the dialog box contains the following sections:

- The edit pane at the top of the dialog box allows you to edit the current expression.
- The toolbar in the middle of the dialog box contains commonly used expression building blocks.
- In the lower half of the dialog box, the left pane is the Selection pane. It displays the folders that are appropriate for the dialog box from which you accessed the expression builder.
- The lower middle pane is the Categories pane. It displays the available categories for the folder you select in the Selection pane. The Search field below the middle pane allows you to search for a value in the middle pane.
- The lower right pane is the Building Blocks pane. It displays the individual building blocks for the category you select in the Category pane. The Search field below the right pane allows you to search for a value in the right pane.

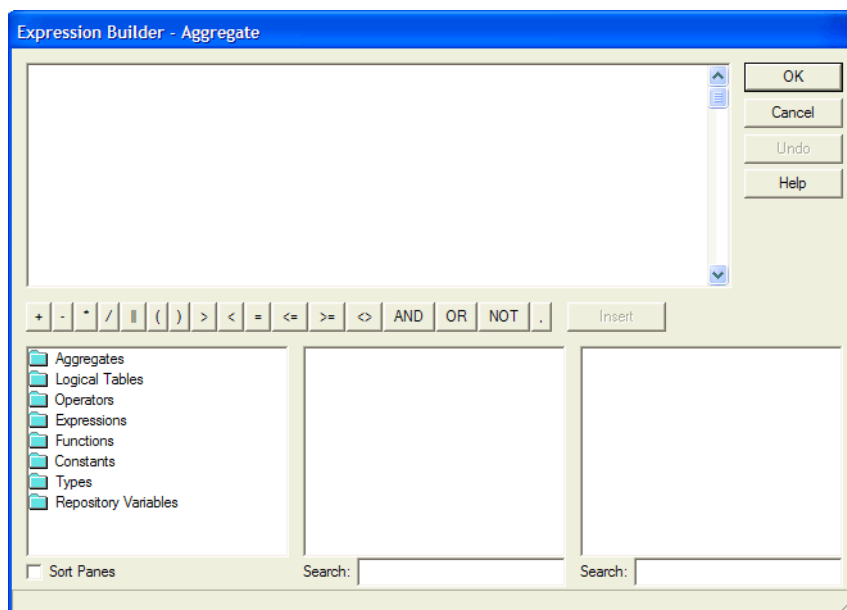


Figure 21. Example Expression Builder

Expression Builder Toolbar

The toolbar is located in the middle portion of the expression builder. [Table 30](#) describes each icon and its function in an expression.

Table 30. Expression Builder Toolbar

Operator	Description
+	Plus sign for addition.
-	Minus sign for subtraction.
*	Multiply sign for multiplication.
/	Divide by sign for division.
	Character string concatenation.
(Open parenthesis.
)	Close parenthesis.
>	Greater than sign, indicating values higher than the comparison.
<	Less than sign, indicating values lower than the comparison.
=	Equal sign, indicating the same value.
<=	Less than or equal to sign, indicating values the same or lower than the comparison.
>=	Greater than or equal to sign, indicating values the same or higher than the comparison.
<>	Not equal to, indicating values higher or lower, but not the same.
AND	AND connective, indicating intersection with one or more conditions to form a compound condition.
OR	OR connective, indicating the union with one or more conditions to form a compound condition.
NOT	NOT connective, indicating a condition is not met.
,	Comma, used to separate elements in a list.

Folders in the Selection Pane

The folders that appear in the Selection pane vary based on the dialog box from which you accessed the expression builder. This section describes the folders that may appear.

Aggregate Content

The Aggregate Content folder contains the available aggregate functions. Aggregate sources must use one of the functions listed here to specify the level of their content.

Dimensions

The Dimensions folder contains the dimension configured in the business model. If no dimension exists in a business model, or if the dimension folder is not pertinent to a particular expression builder, the Dimension folder is not displayed.

When you select the Dimensions folder, each configured dimension displays in the middle pane, and each level for the selected dimension displays in the right pane.

Logical Tables

The Logical Tables folder contains the logical tables configured in the business model. If logical tables are not pertinent to a particular expression builder, the Logical Tables folder is not displayed.

When you select the Logical Tables folder, each logical table in the business model displays in the middle pane, and each column for the selected logical table displays in the right pane.

Operators

The Operators folder contains the available SQL logical operators.

Expressions

The Expressions folder contains the available expressions.

Functions

The Functions folder contains the available functions. The functions that appear depend on the object you selected.

Constants

The Constants folder contains the available constants.

Types

The Types folder contains the available data types.

Repository Variables

This folder contains the available repository variables. If no repository variables are defined, this folder does not appear.

Session Variables

This folder contains the available system session and non system session variables. If no session variables are defined, this folder does not appear.

Example of Setting Up an Expression

Figure 22 shows the expression builder for a derived logical column, with a blank expression.

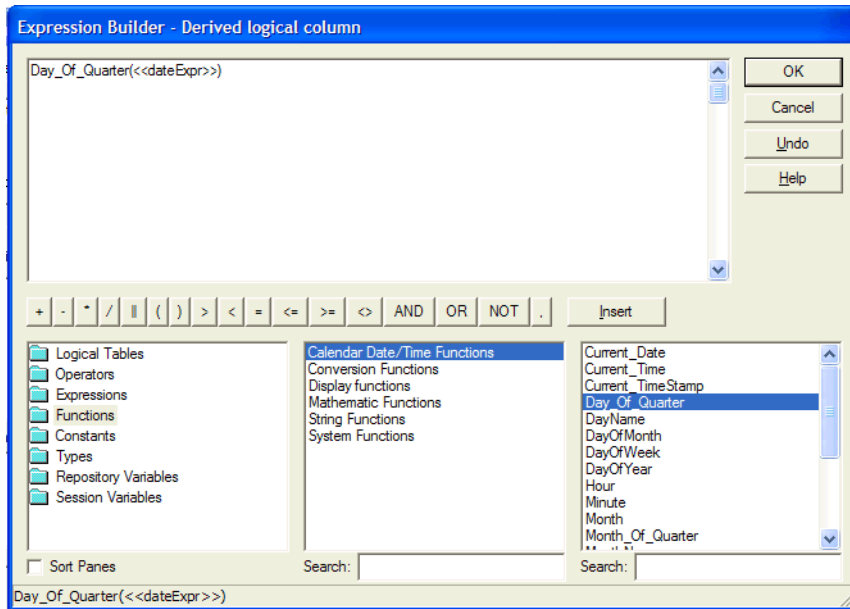


Figure 22. Expression Builder for Derived Logical Columns

With the Functions folder selected in the left pane, double-clicking on the function in the right pane pastes the function in the expression builder's edit box. In the expression builder's edit box, clicking once between the parentheses of the function selects the area, marking it as the insertion point for your next step, adding the argument of the function.

Double-clicking on the logical column pastes the logical column at the insertion point as the argument of the function. [Figure 23](#) shows where the expression appears in the window.

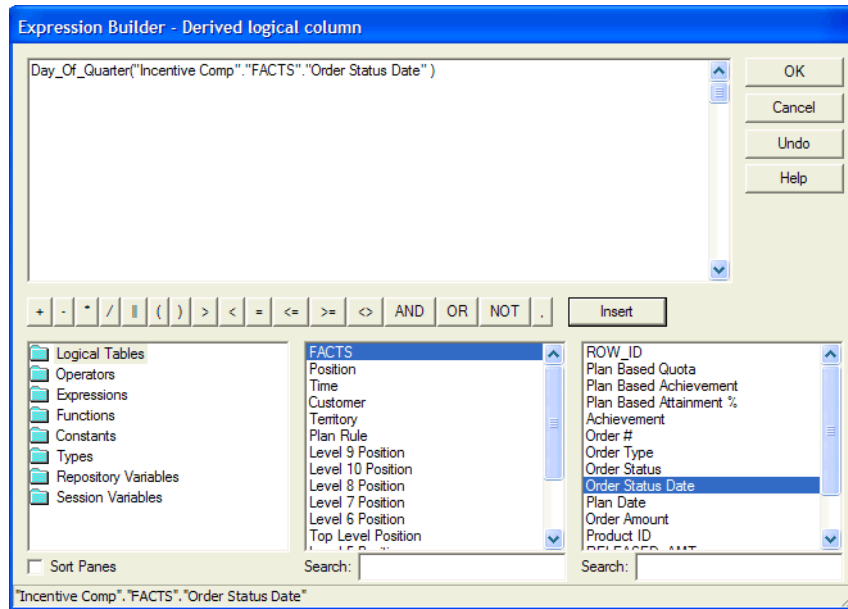


Figure 23. Example Logical Column Function in the Editing Pane

Navigating Within the Expression Builder

Use the following procedure to navigate within an Expression Builder dialog box.

To navigate within an Expression Builder

- 1** In the Selection pane, select the appropriate folder for the type of expression you want to build.
The available categories for the folder appear in the Categories pane.
- 2** Select the appropriate category for the expression you want to build.
The available building blocks for that category appear in the Building Blocks pane.
- 3** Double-click a building block to move it into the Editing pane.
- 4** To insert an operator into the expression, double-click an operator on the Expression Builder toolbar.

Building an Expression

Use this procedure to build an expression in the Expression Builder dialog box.

To build an expression

- 1 Navigate to the individual building blocks you want in the expression.

The Syntax bar at the bottom of the Expression Builder dialog box shows the syntax for the expression.

- 2 Add the building blocks to the Editing pane.
- 3 Edit the building blocks to reflect the expression you want.
- 4 Use the Expression Builder toolbar to insert operators into the expression.
- 5 Repeat the preceding steps until the expression is complete, and then click OK.

The Administration Tool displays a message for any syntax errors in the expression. When the expression is syntactically correct, the Administration Tool adds the expression to the dialog box from which you accessed the Expression Builder.

10 Setting Up Fragmentation Content for Aggregate Navigation

This section contains the following topics:

- [About Aggregate Navigation on page 227](#)
- [Specify Fragmentation Content on page 227](#)

About Aggregate Navigation

Aggregate tables store precomputed results, that are measures that have been aggregated (typically summed) over a set of dimensional attributes. Using aggregate tables is a commonly-used technique for speeding up query response times in decision support systems. This section includes a description of how you can use aggregate navigation and provides setup instructions.

If you are writing SQL queries or using a tool that only understands what physical tables exist (and not their meaning), taking advantage of aggregate tables and putting them to good use becomes more difficult as the number of aggregate tables increases.

Each aggregate table column contains data at a given set of levels. For example, a monthly sales table might contain a precomputed sum of the revenue for each product in each store during each month. You configure this metadata in the Logical Table Source dialog box. For detailed steps showing how to create a logical source, see [“Creating and Administering Logical Table Sources \(Mappings\)” on page 111](#).

Specify Fragmentation Content

When a logical table source does not contain the entire set of data at a given level, you need to specify the portion, or *fragment*, of the set that it does contain. Describe the content in terms of logical columns, using the Fragmentation Content edit box on the Content tab of the Logical Table Source window.

The following examples illustrate techniques and rules for specifying the fragmentation content of sources.

Single Column, Value-Based Predicates

The IN predicates can be replaced with either an equality predicate or multiple equality predicates separated by the OR connective.

Fragment 1:

Logical Column IN <valueList₁>

Fragment *n*:

`Logical Column IN <valueListN>`

Single Column, Range-Based Predicates

Fragment 1:

`Logical Column >= valueof(START_VALUE) AND Logical Column < valueof(MID_VALUE1)`

Fragment 2:

`Logical Column >= valueof(MID_VALUE1) AND Logical Column < valueof(MID_VALUE2)`

Fragment *n*:

`Logical Column >= valueof(MID_VALUEN-1) AND Logical Column < valueof(END_VALUE)`

Pick your start point, midpoints, and endpoint carefully.

NOTE: Notice the use of `>=` and `<` predicates to make sure the fragment content descriptions do not overlap. For each fragment, the upper value needs to be expressed as `<`. You will get an error if you use `<=`. Likewise, you cannot use the `BETWEEN` predicate to describe fragment range content.

The `valueof` referenced here is the value of a repository variable. (For more information about variables, see [Chapter 14, "Using Variables in the Analytics Repository."](#)) If you use repository values in your expression, note that the following construct will not work for Fragment 2:

`Logical Column >= valueof(MID_VALUE1)+1 AND Logical Column < valueof(MID_VALUE2)`

Use another repository variable instead of `valueof(MID_VALUE1)+1`.

The same variables, for example, `valueof(MID_VALUE1)`, do not have to appear in the content of both fragments. You could set another variable, and create statements of the following form:

Fragment 1:

`Logical Column >= valueof(START_VALUE) AND Logical Column < valueof(MID_VALUE1)`

Fragment 2:

`Logical Column >= valueof(MID_VALUE2) AND Logical Column < valueof(MID_VALUE3)`

Multicolumn Content Descriptions

An arbitrary number of predicates on different columns can be included in each content filter. Each column predicate can be value-based or range-based.

Fragment 1:

`<Logical Column1 predicate> AND <Logical Column2 predicate> ... AND <Logical ColumnM predicate>`

Fragment *n*:

`<Logical Column1 predicate> AND <Logical Column2 predicate> ... AND <Logical ColumnM predicate>`

Ideally, all fragments will have predicates on the same M columns. If there is no predicate constraint on a logical column, the Analytics Server assumes that the fragment contains data for all values in that logical column. For exceptions using the OR predicate, see [“Parallel Content Descriptions” on page 229](#).

Parallel Content Descriptions

Unfortunately, the preceding techniques are still not sufficient to handle dates because of the multiple hierarchical relationships across logical columns, such as year > year month > date; month > year month > date. For example, consider fragments delineated by different points in time, such as year and month. Constraining sufficiently far back on year should be enough to drive the selection of just the historical fragment. The parallel OR technique supports this, as shown in the next example. This example assumes that the snapshot month was April 1, 12:00 a.m. in the year 1999. The relevant OR connectives and predicates are shown in bold text.

Fragment 1 (Historical):

```
Enterpri seModel . Peri od. "Day" < VALUEOF("Snapshot Date") OR
Enterpri seModel . Peri od. MonthCode < VALUEOF("Snapshot Year Month") OR
Enterpri seModel . Peri od. "Year" < VALUEOF("Snapshot Year") OR
Enterpri seModel . Peri od. "Year" = VALUEOF("Snapshot Year") AND
Enterpri seModel . Peri od. "Month in Year" < VALUEOF("Snapshot Month") OR
Enterpri seModel . Peri od. "Year" = VALUEOF("Snapshot Year") AND
Enterpri seModel . Peri od. "Monthname" IN (' Mar', ' Feb', ' Jan')
```

Fragment 2 (Current):

```
Enterpri seModel . Peri od. "Day" >= VALUEOF("Snapshot Date") OR
Enterpri seModel . Peri od. MonthCode >= VALUEOF("Snapshot Year Month") OR
Enterpri seModel . Peri od. "Year" > VALUEOF("Snapshot Year") OR
Enterpri seModel . Peri od. "Year" = VALUEOF("Snapshot Year") AND
Enterpri seModel . Peri od. "Month in Year" >= VALUEOF("Snapshot Month") OR
Enterpri seModel . Peri od. "Year" = VALUEOF("Snapshot Year") AND
Enterpri seModel . Peri od. "Monthname" IN (' Dec', ' Nov', ' Oct', ' Sep', ' Aug', ' Jul ',
' Jun', ' May', ' Apr')
```

If the logical model does not go down to the date level of detail, then omit the predicate on `EnterpriseModel.Period."Day"` in the preceding example.

Note the use of the OR connective to support parallel content description tracks.

Examples and Discussion

In this section, the Track n labels in the examples are shown to make it easier to relate the examples to the discussion that follows. You would not include these labels in the actual fragmentation content statement.

Fragment 1 (Historical):

```
Track 1 EnterpriseModel . Period . "Day" < VALUEOF("Snapshot Date") OR
Track 2 EnterpriseModel . Period . MonthCode < VALUEOF("Snapshot Year Month") OR
Track 3 EnterpriseModel . Period . "Year" < VALUEOF("Snapshot Year") OR
Track 4 EnterpriseModel . Period . "Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel . Period . "Month in Year" < VALUEOF("Snapshot Month") OR
Track 5 EnterpriseModel . Period . "Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel . Period . "Monthname" IN (' Mar' , ' Feb' , ' Jan' )
```

For example, consider the first track on EnterpriseModel.Period."Day". In the historical fragment, the < predicate tells the Analytics Server that any queries that constrain on Day before the Snapshot Date fall within the historical fragment. Conversely, the >= predicate in the current fragment on Day indicates that the current fragment does not contain data before the Snapshot Date.

The second track on MonthCode (for example, 199912) is similar to Day. It uses the < and >= predicates as there is a nonoverlapping delineation on month (because the snapshot date is April 1). The key rule to remember is that each additional parallel track needs to reference a different column set. Common columns may be used, but the overall column set needs to be unique. The Analytics Server uses the column set to select the most appropriate track.

The third track on Year (< in the historical fragment and > in the current fragment) tells the Analytics Server that optimal (single) fragment selections can be made on queries that just constrain on year. For example, a logical query on Year IN (1997, 1998) should only hit the historical fragment. Likewise, a query on Year = 2000 needs to hit only the current fragment. However, a query that hits the year 1999 cannot be answered by the content described in this track, and will therefore hit both fragments, unless additional information can be found in subsequent tracks.

The fourth track describes the fragment set with respect to Year and Month in Year (month integer). Notice the use of the multicolumn content description technique, described previously. Notice the use of < and >= predicates, as there is no ambiguity or overlap with respect to these two columns.

The fifth track describes fragment content in terms of Year and Monthname. It uses the value-based IN predicate technique.

As an embellishment, suppose the snapshot date fell on a specific day within a month; therefore multicolumn content descriptions on just year and month would overlap on the specific snapshot month. To specify this ambiguity, <= and >= predicates are used.

Fragment 1 (Historical):

```
EnterpriseModel . Period . "Day" < VALUEOF("Snapshot Date") OR
EnterpriseModel . Period . MonthCode <= VALUEOF("Snapshot Year Month") OR
EnterpriseModel . Period . "Year" < VALUEOF("Snapshot Year") OR
EnterpriseModel . Period . "Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel . Period . "Month in Year" <= VALUEOF("Snapshot Month") OR
EnterpriseModel . Period . "Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel . Period . "Monthname" IN (' Apr' , ' Mar' , ' Feb' , ' Jan' )
```

Fragment 2 (Current):

```
Enterpri seModel . Peri od. "Day" >= VALUEOF("Snapshot Date") OR
Enterpri seModel . Peri od. MonthCode >= VALUEOF("Snapshot Year Month") OR
Enterpri seModel . Peri od. "Year" > VALUEOF("Snapshot Year") OR
Enterpri seModel . Peri od. "Year" = VALUEOF("Snapshot Year") AND
Enterpri seModel . Peri od. "Month in Year" >= VALUEOF("Snapshot Month") OR
Enterpri seModel . Peri od. "Year" = VALUEOF("Snapshot Year") AND
Enterpri seModel . Peri od. "Monthname" IN ( ' Dec' , ' Nov' , ' Oct' , ' Sep' , ' Aug' , ' Jul' ,
' Jun' , ' May' , ' Apr' )
```

Unbalanced Parallel Content Descriptions

In an order entry application, time-based fragmentation between historical and current fragments is typically insufficient. For example, records may still be volatile, even though they are historical records entered into the database before the snapshot date.

Assume, in the following example, that open orders may be directly updated by the application until the order is shipped or canceled. After the order has shipped, however, the only change that can be made to the order is to type a separate compensating return order transaction.

There are two parallel tracks in the following content descriptions. The first track uses the multicolumn, parallel track techniques described in the preceding section. Note the parentheses nesting the parallel calendar descriptions within the Shipped-or-Canceled order status multicolumn content description.

The second parallel track is present only in the Current fragment and specifies that all Open records are in the Current fragment only.

Fragment 1 (Historical):

```
Marketi ng. "Order Status". "Order Status" IN ( ' Shi pped' , ' Cancel ed' ) AND
Marketi ng. Cal endar. "Cal endar Date" <= VALUEOF("Snapshot Date") OR
Marketi ng. Cal endar. "Year" <= VALUEOF("Snapshot Year") OR
Marketi ng. Cal endar. "Year Month" <= VALUEOF("Snapshot Year Month")
```

Fragment 2 (Current):

```
Marketi ng. "Order Status". "Order Status" IN ( ' Shi pped' , ' Cancel ed' ) AND
Marketi ng. Cal endar. "Cal endar Date" > VALUEOF("Snapshot Date") OR
Marketi ng. Cal endar. "Year" >= VALUEOF("Snapshot Year") OR
Marketi ng. Cal endar. "Year Month" >= VALUEOF("Snapshot Year Month")
OR Marketi ng. "Order Status". "Order Status" = ' Open'
```

The overlapping Year and Month descriptions in the two fragments do not cause a problem, as overlap is permissible when there are parallel tracks. The rule is that at least one of the tracks has to be nonoverlapping. The other tracks can have overlap.

Aggregate Table Fragments

Information at a given level of aggregation is sometimes stored in multiple physical tables. When individual sources at a given level contain information for a portion or fragment of the domain, the Analytics Server needs to know the content of the sources in order to pick the appropriate source for the query.

For example, suppose you have a database that tracks the sales of soft drinks in all stores. The detail level of data is at the store level. Aggregate information, as described in [Figure 24](#), is stored at the city level for the sales of Coke and Pepsi, but there is no aggregate information for the sales of 7-Up or any other of the sodas.

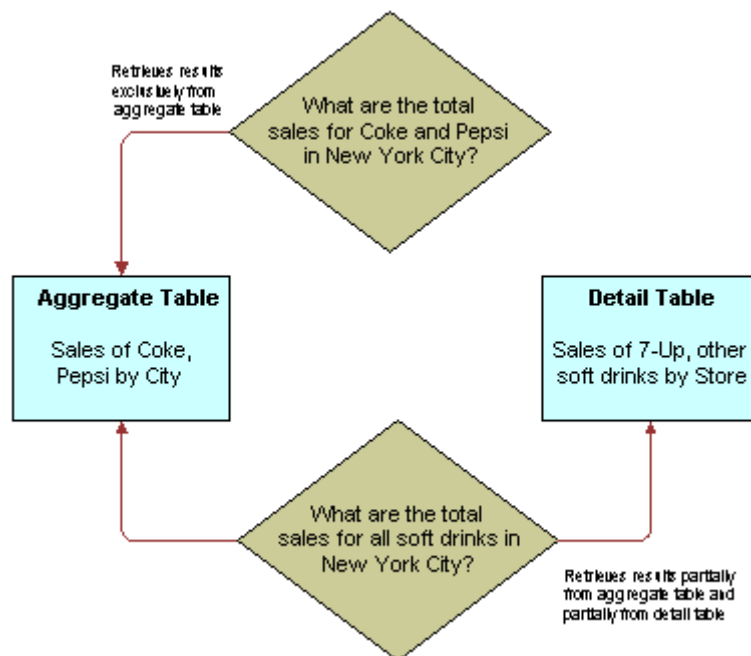


Figure 24. Aggregating Information

The goal of this type of configuration is to maximize the use of the aggregate table. If a query asks for sales figures for Coke and Pepsi, the data should be returned from the aggregate table. If a query asks for sales figures for all soft drinks, the aggregate table should be used for Coke and Pepsi and the detail data for the other brands.

The Analytics Server handles this type of partial aggregate navigation. To configure a repository to use aggregate fragments for queries whose domain spans multiple fragments, you need to define the entire domain for each level of aggregate data, even if you have to configure an aggregate fragment as being based on a less summarized physical source.

Specify the Aggregate Table Content

You configure the aggregate table navigation in the logical table source mappings. In the soft drink example, the aggregate table contains data for Coke and Pepsi sales at the city level. Its Aggregate content specification (in the Content tab of the Logical Table Source window) is similar to the following:

Group by logical level:

```
GeographyDim.CityLevel, ProductDim.ProductLevel
```

Its Fragmentation content specification (also in the Content tab of the Logical Table Source dialog) is similar to the following:

```
SoftDrinks.Products.Product IN ('Coke', 'Pepsi')
```

This content specification tells the Analytics Server that the source table has data at the city and product level for two of the products. Additionally, because this source is a fragment of the data at this level, you need to check the option *This source should be combined with other sources at this level*, in the Content tab of the Logical Table Source dialog box, to indicate that the source combines with other sources at the same level. For more information, see [“Specify Fragmentation Content” on page 227](#).

Define a Physical Layer Table with a Select Statement to Complete the Domain

The data for the rest of the domain (the other types of sodas) is all stored at the store level. To define the entire domain at the aggregate level (city and product, in this example), you need to have a source that contains the rest of the domain at this level. Because the data at the store level is at a lower (that is, more detailed) level than at the city level, it is possible to calculate the city and product level detail from the store and product detail by adding up the product sales data of all of the stores in a city. This can be done in a query involving the store and product level table.

One way to do this is to define a table in the Physical layer with a Select statement that returns the store level calculations. To define the table, create a table in the Physical layer by selecting the physical schema folder that the Select statement will be querying and execute the New Table command. Choose Select from the Object Type drop-down list, and type the SQL statement in the pane to the right.

The SQL needs to define a virtual table that completes the domain at the level of the other aggregate tables. In this case, there is one existing aggregate table, and it contains data for Coke and Pepsi by city. Therefore, the SQL statement has to return all of the data at the city level, except for the Coke and Pepsi data.

Figure 25 shows the Physical Table dialog for this virtual table, along with sample aggregate and detail physical tables definitions:

Physical aggregate table
containing the sales of Coke and
Pepsi by city

CityProductSales
Product
City
Dollars

Physical detail table containing
data for all products by store

StoreSales
Product
Store
Dollars

SELECT statement that
defines a virtual table to
complete the domain of
products by city. Coke and
Pepsi are omitted because
they are defined in the
aggregate table
CityProductSales.
Note the aliases referencing
the aggregate table column.

Figure 25. Example Physical Table Definitions

Specify the SQL Virtual Table Content

Next, create a new logical table source for the Sales column that covers the remainder of the domain at the city and product level. This source contains the virtual table created in the previous section. Map the Dollars logical column to the USDollars physical column in this virtual table.

The Aggregate content specification (in the Content tab of the Logical Table Source dialog) for this source is:

Group by logical level:

GeographyDi m. Ci tyLevel , ProductDi m. ProductLevel

This tells the Analytics Server this source has data at the city and product level.

The Fragmentation content specification might be:

SoftDri nks. Products. Product = ' 7-Up'

Additionally, because it combines with the aggregate table containing the Coke and Pepsi data at the city and product level to complete the domain, you need to check the option in the Content tab of the Logical Table Source dialog indicating that the source is combined with other sources at the same level.

Physical Joins for Virtual Table

Construct the correct physical joins for the virtual table. Notice that CityProductSales2 joins to the Cities and Products tables in [Figure 26](#).

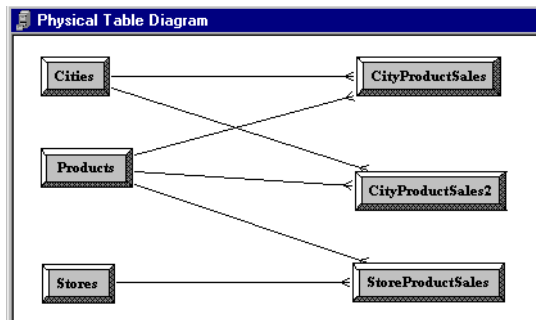


Figure 26. Example Physical Joins

In this example, the two sources comprise the whole domain for soda sales. A domain may have many sources. The sources have to all follow the rule that each level needs to contain sources that, when combined together, comprise the whole domain of values at that level. Setting up the entire domain for each level helps to make sure that queries asking for Coke, Pepsi, and 7-Up do not leave out 7-Up. It also helps to make sure that queries requesting information that has been precomputed and stored in aggregate tables can retrieve that information from the aggregate tables, even if the query requests other information that is not stored in the aggregate tables.

11 Administering the Query Environment

The Analytics Server is a server-based query environment and has many of the tools associated with managing server-based systems. This section describes how to use these tools to perform various administrative actions, including starting and stopping the server, checking and analyzing the log files, and other tasks related to managing a multiuser environment.

This chapter contains the following topics:

- [Starting the Analytics Server \(Siebel Analytics Server Service\) on page 237](#)
- [Shutting Down the Analytics Server on page 240](#)
- [Getting Users to Connect to the Server on page 242](#)
- [Administering the Query Log on page 242](#)
- [Administering Usage Tracking on page 246](#)
- [Server Session Management on page 253](#)
- [Server Configuration and Tuning on page 255](#)

Starting the Analytics Server (Siebel Analytics Server Service)

The Analytics Server needs to be running before any queries can be processed. The following topics describe three ways to start the Analytics Server, how to change the Analytics Server User ID, and what to do if the Analytics Server fails to start:

- [Starting the Server from Windows Services on page 237](#)
- [Configuring the Server for Automatic Startup in Windows on page 238](#)
- [Running the Analytics Server Startup Script in UNIX on page 238](#)
- [Changing the User ID in Which the Analytics Server Runs on page 239](#)
- [If the Server Fails to Start on page 240](#)

Starting the Server from Windows Services

This procedure requires your user ID to be a member of the Windows Administrators group on the local machine on which the Analytics Server is installed.

To start the Analytics Server (Siebel Analytics Server service)

- 1 On the machine in which the server is installed, choose Start > Settings > Control Panel > Services.

You might need to open the Administrative Tools submenu.

- 2 Select the Siebel Analytics Server service and click Start.

A message appears indicating that the service is starting. It might take a few minutes for the server to start because it loads all repositories listed in the Repositories section of the NQConfig.INI file during the startup operation.

When startup is complete, the startup message goes away and the status of the service changes to Started. The following information is logged to the NQServer.log file, located in the Log subdirectory of the Siebel Analytics installation folder:

- Startup time
- Any business models that are loaded
- Any errors that occurred

If startup does not complete, check to make sure that there are no errors in the NQConfig.INI file, such as the incorrect spelling of a repository filename. If you receive an informational message stating the server has not yet started, refresh the status periodically until the status changes to Started.

Configuring the Server for Automatic Startup in Windows

The following procedure explains how to configure the Analytics Server to start automatically when Windows NT or Windows 2000 starts.

To configure the Analytics Server (Siebel Analytics Server service) for automatic startup

- 1 On the machine in which the server is installed, choose Start > Settings > Control Panel > Services.

You might need to open the Administrative Tools submenu.

- 2 In the Services dialog box, double-click the Siebel Analytics Server service.
- 3 In the Siebel Analytics Server Properties dialog box, from the Startup Type drop-down list, select Automatic, and then click OK.

The Siebel Analytics Server service will now start automatically when Windows starts.

Running the Analytics Server Startup Script in UNIX

Start the Analytics Server by running one of the following scripts:

- If you are using sh or bash:

```
run-sa.sh start
```

- If you are using csh:

```
run-sa.csh start
```

- If you have set up your environment with sa.sh or sa.csh:

```
nqscmgateway.exe &
```

- For the standard shell:

```
nohup nqscmgateway.exe >/dev/null 2>&1 &
```

- For the C shell:

```
nohup nqscmgateway.exe >&/dev/null &
```

Changing the User ID in Which the Analytics Server Runs

In Windows, the Analytics Server (Siebel Analytics Server) runs as a Windows service. It runs under the local system account by default. If the server needs to access databases on remote machines, it needs to run under a user ID that has the appropriate network privileges. Additionally, the user ID has to be a member of the Windows NT Administrators group on the local machine.

If you want to change the user ID in which the Analytics Server runs, you can do so from the Control Panel Services utility.

To change the user ID

- 1 On the machine in which the Analytics Server is installed, select Start > Settings > Control Panel.
- 2 In the Control Panel, double-click the Services icon.
- 3 Double-click the Siebel Analytics Server service and click Startup.
- 4 In the Siebel Analytics Server Properties dialog box, in the Log On As area, select This Account, and then click the button to the right of the text box.
- 5 In the Add User dialog box, select the user account in which you want the Analytics Server to run, click Add, and then click OK.
- 6 Type the password for the user in the Services dialog box, confirm the password, and then click OK.

The server is now configured to run under the new user ID. The next time you start the service, it will attempt to use the new account to start the service.

If the Server Fails to Start

If the startup operation fails, look in the following log files for messages indicating why:

- In Windows NT and Windows 2000, in the Windows Event log, that you can access by selecting Start > Programs > Administrative Tools > Event Viewer.
- In Windows NT, Windows 2000, and UNIX, in the NQServer.log file. This file is located in the Log folder in the Analytics Server software installation folder (\Siebel Analytics). You can use a text editor to view this file.
- In UNIX, run /usr/sbin/syslogd and look for any system and Analytics Server-related messages.

The log files contain messages indicating why the server startup failed. For example, if there were a syntax error in the NQConfig.INI file, both the operating system's log and the NQServer.log file would contain messages about the syntax error. After examining the log messages, correct the problem and start the server again.

Shutting Down the Analytics Server

You can stop the Analytics Server (Siebel Analytics Server) in any of the following ways.

- [Shutting Down the Server in Windows Services on page 240](#)
- [Shutting Down the Server from a Command Prompt in Windows on page 241](#)
- [Running the Analytics Server Shutdown Script in UNIX on page 241](#)
- [Shutting Down the Analytics Server Using the Administration Tool on page 242](#)

When you shut down the server, any active client connections receive an error message and are immediately terminated, and any outstanding queries to underlying databases are cancelled.

Shutting Down the Server in Windows Services

The following procedure explains how to shut down the Analytics Server from the Windows Control Panel Services applet.

To shut down the server from the Services applet

- 1 On the machine in which the Analytics Server is installed, select Start > Settings > Control Panel.
- 2 Open the Services applet by double-clicking the Services icon in the Control Panel.
- 3 Select the Siebel Analytics Server service and click Stop.

A message appears indicating that the service is shutting down.

When shutdown is complete, the status in the Services Control Panel becomes blank and a log message is written to the NQServer.log file, located in the Log folder in the software installation folder (\Siebel Analytics\Log).

Shutting Down the Server from a Command Prompt in Windows

Use this procedure in Windows to shut down the Analytics Server from a Command prompt.

To shut down the Analytics Server from a Windows command prompt

- Type the following command in the machine in which the Analytics Server is running:

```
nqsshutdown {-d <data_source_name> -u <user ID> -p <password>}
```

where:

data_source_name The name of a nonclustered ODBC data source used to connect to the server to perform the shut down operation. The data source needs to connect to the server as an analytics user who is a member of the Analytics Server Administrators group. Only users defined as Analytics Server administrators can shut down the server.

NOTE: The `nqsshutdown` command is not valid for clustered DSNs. When passed a standard (nonclustered) DSN, it will shut down the targeted Analytics Server even if the server is participating in a cluster.

user ID The user to connect to the Analytics Server to perform the shut down operation.

password The password for the user ID connecting to the Analytics Server to perform the shut down operation.

When shutdown is complete, a message indicating this is displayed in the Command window.

Running the Analytics Server Shutdown Script in UNIX

Stop the Analytics Server by running one of the following scripts:

- If you are using `sh` or `bash`:

```
run-sa.sh stop
```

- If you are using `csh`:

```
run-sa.csh stop
```

- If you have set up your environment with `sa.sh` or `sa.csh`:

```
nqsshutdown.exe -uAdministrator
```

Shutting Down the Analytics Server Using the Administration Tool

The following procedure explains how to shut down the server using the Administration Tool. You need to open the repository in online mode using a nonclustered DSN.

NOTE: To shut down the server, the DSN has to log in as a user that has administrator authority.

To shut down the server using the Administration Tool

- 1 Start the Administration Tool by selecting Start > Programs > Siebel Analytics > Siebel Analytics Administration Tool.
- 2 Open a repository that is loaded on the server in online mode.
- 3 Select File > Shut Down Server.
- 4 When a dialog box appears asking you to confirm the shutdown, click Yes.

This shuts the server down and ends the Administration Tool online session. When connected using a clustered DSN, use the Cluster Manager to take individual Analytics Server instances offline. For more information, see [“Using the Cluster Manager” on page 318](#).

Getting Users to Connect to the Server

Users need to set up a data source to connect to a business model within an analytics repository. For information about setting up DSN connections, see [“Configuring Analytics ODBC Data Source Names \(DSNs\)” on page 279](#).

You can also run the Administration Tool from a remote machine in online mode or offline mode.

Administering the Query Log

The Analytics Server provides a facility for logging query activity at the individual user level. The query log file is named the NQQuery.log file. This file is in the Log subdirectory in the Siebel Analytics installation folder. Logging should be used for quality assurance testing, debugging, and troubleshooting by Oracle's Siebel Technical Support. In production mode, query logging is typically disabled.

Analytics Server query logging is tracked at a user level. It will be a resource intensive process if you track the entire user community. For production systems, it is recommended that it be enabled for a very targeted user community only. Most users should have a log level of 0 (zero).

It is recommended that you only test users when the user name clearly indicates it is a test user and have verified that query logging enabled. If logging is enabled for such users, it is recommended that they be given names such as sales_admin_with_logging, sales_dev_with_logging, or sales_test_with_logging, so that you can readily identify them. Even production administrator logins should not have query logging enabled because it could strain the available resources.

You should also disable query logging for the following:

- The SQL statement in the initialization string. The Initialization string text box is in the Initialization Block dialog box, in the General tab.
- NOTE:** The LOGGING column references stored values for the log level.
- The default initializer value should be set to 0 (zero) for the LOGLEVEL session variable. The default initializer field is in the Initialization Block dialog box, in the Variables tab.
 - The logging level should be set to 0 (zero) for each production user. The Logging level field is in the User dialog box, in the User tab.

Configuring the Logging System

This section describes the logging system and includes information about setting the size of the query log, choosing a logging level, and enabling query logging for a user.

Because query logging can produce very large log files, the logging system is turned off by default. It is sometimes useful, however, to enable logging to test that your repository is configured properly, to monitor activity on your system, to help solve performance problems, or to assist Technical Support. You need to enable logging on the system for each user whose queries you want logged.

Controlling the Size of the Log File

The parameter USER_LOG_FILE_SIZE in the User Log section of the NQSSConfig.INI file determines the size of the NQQuery.log file. When the log file grows to one-half the size specified by the USER_LOG_FILE_SIZE parameter, the file is renamed to NQQuery.log.old, and a new log file is created automatically. (This helps to make sure that the disk space allocated for the log file does not exceed the size specified in the configuration file.) Only one copy of the old file is kept.

You can set the file size as high as you like, limited only by the amount of space available on the device. If you change the value of the USER_LOG_FILE_SIZE parameter, you need to restart the Analytics Server (Siebel Analytics Server) for the change to take effect. For the syntax of the USER_LOG_FILE_SIZE parameter, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Setting a Logging Level

You can enable logging level for individual users; you cannot configure a logging level for a group.

NOTE: A session variable overrides a user's logging level. For example, if the administrator has a logging level defined as 4 and a session variable logging level is defined as default 0 (zero) in the repository, the administrator's logging level will be 0.

Set the logging level based on the amount of logging you want to do. In normal operations, logging is generally disabled (the logging level is set to 0). If you decide to enable logging, choose a logging level of 1 or 2. These two levels are designed for use by administrators.

NOTE: Logging levels greater than 2 should be used only with the assistance of Technical Support.

The logging levels are described in [Table 31](#).

Table 31. Logging Levels

Logging Level	Information That Is Logged
Level 0	No logging.
Level 1	<p>Logs the SQL statement issued from the client application.</p> <p>Logs elapsed times for query compilation, query execution, query cache processing, and backend database processing.</p> <p>Logs the query status (success, failure, termination, or timeout). Logs the user ID, session ID, and request ID for each query.</p>
Level 2	<p>Logs everything logged in Level 1.</p> <p>Additionally, for each query, logs the repository name, business model name, presentation catalog name, SQL for the queries issued against physical databases, queries issued against the cache, number of rows returned from each query against a physical database and from queries issued against the cache, and the number of rows returned to the client application.</p>
Level 3	<p>Logs everything logged in Level 2.</p> <p>Additionally, logs the logical query plan. Do not select this level without the assistance of Technical Support.</p>
Level 4	<p>Logs everything logged in Level 3.</p> <p>Additionally, logs the query execution plan. Do not select this level without the assistance of Technical Support.</p>
Level 5	<p>Logs everything logged in Level 4.</p> <p>Additionally, logs intermediate row counts at various points in the execution plan. Do not select this level without the assistance of Technical Support.</p>
Level 6 and 7	Reserved for future use.

To set a user's logging level

- 1 In the Administration Tool, select Manage > Security.
The Security Manager dialog box appears.
- 2 Double-click the user's user ID.
The User dialog box appears.
- 3 Set the logging level by clicking the Up or Down arrows next to the Logging Level field.

To disable a user's logging level

- Set the logging level to 0.

Using the Log Viewer

Use the Siebel Business Analytics log viewer utility nQLogViewer (or a text editor) to view the query log. Each entry in the query log is tagged with the user ID of the user who issued the query, the session ID of the session in which the query was initiated, and the request ID of the individual query.

To run the nQLogViewer utility, open a Command window and type nQLogViewer with any combination of its arguments. The syntax is as follows:

```
nqlogviewer [-u<user_ID>] [-f<log_input_filename>]
            [-o<output_result_filename>]
            [-s<session_ID>] [-r<request_ID>]
```

where:

<i>user_ID</i>	The name of a user in the analytics repository. This limits the scope to entries for a particular user. If not specified, all users for whom query logging is enabled are shown.
<i>log_input_filename</i>	The name of an existing log file. This parameter is required.
<i>output_result_filename</i>	The name of a file in which to store the output of the log viewer. If the file exists, results are appended to the file. If the file does not exist, a new file is created. If not specified, output is sent to the monitor screen.
<i>session_ID</i>	The session ID of the user session. The Analytics Server assigns each session a unique ID when the session is initiated. This limits the scope of the log entries to the specified session ID. If not specified, all session IDs are shown.
<i>request_ID</i>	The request ID of an individual query. The Analytics Server assigns each query a unique ID when the query is initiated. This limits the scope of the log entries to the specified request ID. If not specified, all request IDs are shown.

You can also locate user IDs, session IDs and request IDs through the Session Manager. For more information, see [“Using the Session Manager” on page 253](#).

NOTE: Web administrators can view the query log using the [Manage Sessions Web page](#).

Interpreting the Log Records

After you have logged some query information and started the log viewer, you can analyze the log. The log is divided into several sections, some of which are described in the next section. Log entries for levels 1 and 2 are generally self-explanatory. The log entries can provide insights to help DBAs in charge of the underlying databases tune them for optimum query performance. The query log can also help you check the accuracy of applications that use the Analytics Server.

SQL Request

This section lists the SQL issued from the client application. This can be used to rerun the query from the same application, or from a different application.

General Query Information

This section lists the repository, the business model, and the presentation catalog from which the query was run. You can use this information to provide statistics on query usage that could be used to set priorities for future application development and system management.

Database Query

This section of the log begins with an entry that reads Sending query to the database named <data_source_name>, where data_source_name is the name of the data source to which the Analytics Server is connecting. Multiple database queries can be sent to one or more data sources. Each query will have an entry in the log.

The database query section has several uses. It records the SQL sent to the underlying databases; you can then use the logged SQL to run queries directly against the database for performance tuning, results verification, or other testing purposes. It allows you to examine the tables that are being queried to verify that aggregate navigation is working as you expect. If you understand the structure of the underlying database, it might also provide some insights into potential performance improvements, such as useful aggregate tables or indexes to build.

Query Status

The query success entry in the log indicates if the query completed successfully or if it failed. You can search through the log for failed queries to determine why they failed. For example, all the queries during a particular time period might have failed due to a database downtime.

Administering Usage Tracking

The Analytics Server supports the accumulation of usage tracking statistics that can be used in a variety of ways—for example, database optimization, aggregation strategies, or billing users or departments based on the resources they consume. The Analytics Server tracks usage at the detailed query level.

When you enable usage tracking, statistics for every query are inserted into a database table or are written to a usage tracking log file. If you use direct insertion, the Analytics Server directly inserts the usage tracking data into a relational database table. It is recommended that you use direct insertion to write statistics to a database table.

This section contains the following topics:

- [Setting Up Direct Insertion to Collect Information for Usage Tracking on page 247](#)
- [Setting Up a Log File to Collect Information for Usage Tracking on page 249](#)

Setting Up Direct Insertion to Collect Information for Usage Tracking

This is the recommended method for setting up usage tracking.

To set up direct insertion for usage tracking, use the guidelines in this section. For more information, see *Siebel Business Analytics Platform Installation and Configuration Guide*. To set up and administer direct insertion, use the following topics:

- [Enabling Direct Insertion on page 247](#)
- [Database Table Configuration on page 247](#)
- [Connection Pool Configuration on page 248](#)
- [Buffer Size Configuration Parameter on page 248](#)
- [Buffer Time Limit Configuration Parameter on page 248](#)
- [Number of Insert Threads Configuration Parameter on page 248](#)
- [Max Inserts Per Transactions Configuration Parameter on page 248](#)

Enabling Direct Insertion

In the Usage Tracking section of the NOSConfig.ini file, the DIRECT_INSERT parameter determines whether the query statistics are inserted directly into a database table or are written to a file for subsequent loading. The DIRECT_INSERT and ENABLE parameters must be set to YES to enable direct insertion.

NOTE: It is recommended to enable direct insertion.

Database Table Configuration

Inserting query statistic information into a table requires the configuration of the name of the table and the connection pool used to access the table.

The fully qualified physical table name consists of up to four components (database name, catalog name, schema name, and table name). Each component is surrounded by double quotes (") and separated by a period (.). The physical table name must be fully qualified. This fully qualified physical table name must match a table name in the physical layer of the loaded repository. The following is an example of a physical table name for the Usage Tracking table in the analytics repository:

```
PHYSICAL_TABLE_NAME = "Siebel Analytics Usage"."Catalog"."dbo"."S_NQ_ACCT" ;
```

In this example, Siebel Analytics Usage represents the database component, Catalog represents the catalog component, dbo represents the schema component, and S_NQ_ACCT represents the table name.

Connection Pool Configuration

The fully-specified connection pool name has two parts, database name and connection pool name. Each part is surrounded by double quotes (") and separated by a period (.). The fully qualified connection pool name should match a connection pool name in the physical layer of the loaded repository. For an example, see the following connection pool name in the analytics repository:

```
CONNECTION_POOL = "Siebel Analytics Usage"."Connection Pool" ;
```

In this example, Siebel Analytics Usage represents the database component and Connection Pool represents the connection pool name proper.

For Usage Tracking inserts to succeed, the connection pool must be configured with a user ID that has write access to the back-end database.

NOTE: It is recommended that the connectivity type supports international data.

Buffer Size Configuration Parameter

The BUFFER_SIZE configuration parameter indicates how much memory the Analytics Server should allocate for buffering the insert statements. Such a buffer allows the Analytics Server to submit multiple insert statements as part of a single transaction, improving Usage Tracking insert throughput. It also means that ordinary query requests do not have to wait on Usage Tracking inserts, improving average query response time. You may want to adjust this value based on available memory and memory utilization on the server machine.

Buffer Time Limit Configuration Parameter

The BUFFER_TIME_LIMIT_SECONDS configuration parameter indicates the maximum amount of time an insert statement will remain in the buffer before the Usage Tracking subsystem attempts to issue it. This time limit ensures that the Analytics Server will issue the insert statements in a timely manner even during periods of extended quiescence.

Number of Insert Threads Configuration Parameter

The NUM_INSERT_THREADS configuration parameter indicates the number of threads that will remove insert statements from the buffer and issue them to the Usage Tracking database. Assuming separate connection pools for readers and inserters, the number of insert threads should typically equal the Maximum Connections setting in the connection pool.

Max Inserts Per Transactions Configuration Parameter

The MAX_INSERTS_PER_TRANSACTION configuration parameter indicates the maximum number of insert statements the Usage Tracking subsystem attempts to issue as part of a single transaction. The larger this number, the greater potential throughput for Usage Tracking inserts. However a larger number also increases the likelihood of transactions failing due to deadlocks. Note that a small value for BUFFER_TIME_LIMIT_SECONDS may limit the number of inserts per transaction.

Setting Up a Log File to Collect Information for Usage Tracking

This is an alternate method for setting up usage tracking. It is recommended that you use direct insertion to collect information for usage tracking. For more information, see [“Setting Up Direct Insertion to Collect Information for Usage Tracking” on page 247](#).

This section contains the following topics:

- [Selecting an Output Location on page 249](#)
- [File Naming Conventions on page 250](#)
- [Output File Format on page 250](#)
- [Performance Considerations on page 252](#)

Selecting an Output Location

The parameter `STORAGE_DIRECTORY` in the Usage Tracking section of the `NQConfig.INI` file determines the location of usage tracking log files. If usage tracking is enabled, but no storage folder is specified, the files are written in the Log folder in the software installation folder (\\Siebel Analytics).

Current files are periodically written to disk, and new files are created. The parameter `CHECKPOINT_INTERVAL_MINUTES` controls the frequency with which usage tracking data is flushed to disk, and the parameter `FILE_ROLLOVER_INTERVAL_MINUTES` controls the frequency with which the current usage tracking log file is closed and a new file created.

When usage tracking is enabled, every query is logged to a usage tracking log file. This may require a large amount of available storage. For example, assume an average of 300 bytes of data output for each query and 10 queries per second over an 8 hour day. This results in approximately 83 MB of usage tracking data written to storage per day. If this example is extended to a 24 x 7 operation, the result is approximately .25 GB of storage per day.

The Analytics Server has no limit on the size or quantity of usage tracking log files that can exist in the specified location. It is the responsibility of the user to make sure that sufficient space is available, and to remove or archive old usage tracking files.

NOTE: Insufficient storage space may cause you to lose usage tracking data. If the Analytics Server encounters an error accessing a usage tracking output file, it immediately discontinues the collection of usage tracking statistics and issues an error message to the `NQServer.log` and, in Windows, to the Windows Event log. Even if additional storage space is made available, the collection of usage tracking statistics will not resume until the server is restarted.

File Naming Conventions

The file naming scheme for the usage tracking log files is NQAcct.yyyymmdd.hhmmss.log, where yyyy is the year, mm is the month, dd is the day, hh is the hour, mm is the minute, and ss is the second of the timestamp when the file was created. For example, if the server creates the usage tracking log file at 07:15:00 AM on February 12, 2003, the filename would be NQAcct.20030212.071500.log. After the specified rollover interval, this file is flushed to disk and closed and a new log file, with the current date and timestamp, is created.

Output File Format

The usage tracking log files are text files, in semicolon-delimited (;) format.

(A semicolon is used as the column delimiter because the logical SQL text contains commas.) A line feed delimits the end of each row of data.

The schema is described in [Table 32](#). For more information about the contents of each column, see [“Description of the Usage Tracking Data” on page 414](#).

Table 32. Usage Tracking Output File Format

Column Number	Column Name	Data Type	Max Data Size	Nullable
1	User name	Varchar	128	No
2	Repository name	Varchar	128	No
3	Subject area name	Varchar	128	No
4	Node ID	Varchar	15	No
5	Start timestamp	Char (Timestamp)	19	No
6	Start date	Char (yyyy-mm-dd)	10	No
7	Start hourMin	Char (hh:mm)	5	No
8	End timestamp	Char (Timestamp)	19	No
9	End date	Char (yyyy-mm-dd)	10	No
10	End hourMin	Char (hh:mm)	5	No
11	Query Text	Varchar	1024	No
12	Success indicator	Integer (see following Note)	4	No
13	Row count	Integer (see following Note)	4	Yes
14	Total time (secs)	Integer (see following Note)	4	Yes
15	Compilation time (secs)	Integer (see following Note)	4	Yes
16	Number of database queries	Integer (see following Note)	4	Yes
17	Cumulative db time (secs)	Integer (see following Note)	4	Yes
18	Cumulative db rows	Integer (see following Note)	4	Yes
19	Cache indicator	Char	1	No
20	Query source	Varchar	30	No
21	Analytics Web catalog path	Varchar	250	No
22	Analytics Web dashboard name	Varchar	150	Yes

NOTE: All data in the output file is in character format. The data in columns 12 through 18 are output as text representations of integer numbers. Therefore, they behave more like Varchar(10) columns than integers. For example, if the row count is one million rows, then 1000000 appears in the output file in column 13 (Row count). This constitutes seven bytes of data, even though the data represents a 4-byte internal integer value.

- In column 12, a Success indicator value of 0 signifies a successful query. All nonzero values indicate failure. The following failure indicators are currently defined:
 - 1 indicates timeout
 - 2 indicates row limit violation
 - 3 indicates unknown error

The subsequent columns are valid only if the Success indicator signifies a successful query (value is 0):

- The Start timestamp and End timestamp columns indicate the wall clock time when the logical query started and finished. Each value is 19 bytes of character data representing a SQL-92 timestamp. The format is *yyyy-mm-dd-hh:mm:ss*. The related columns, Start date and End date, contain just the date component from the respective timestamps (in the *yyyy-mm-dd* format). Finally, the related columns, Start hourMin and End hourMin, contain just the hour and minute components from the respective timestamps (in a char *hh:mm* format).

While there is no guaranteed unique key for the usage tracking data, a combination of User name, Node ID, Start timestamp and Query text will usually be sufficient.

For information about sample scripts to help you extract data from usage tracking log files and load it to appropriately formatted relational database tables, see [“Usage Tracking Data Descriptions and Using the Log File Method” on page 413](#).

Performance Considerations

When usage tracking is enabled, the Analytics Server collects usage tracking data for every query. This data, however, is only written to disk at user-specified intervals, known as *checkpoints*. The default setting is to checkpoint every 5 minutes.

While this value can be modified in the `theNQSConfig.INI` file (see *Siebel Business Analytics Platform Installation and Configuration Guide*), reducing the interval adds overhead and, if set low enough, could potentially impact server performance. Setting the value higher increases the amount of usage tracking data that could be lost in the unlikely event of an abnormal shutdown of the Analytics Server.

The Analytics Server periodically initiates usage tracking log file rollovers. A *rollover* consists of closing the current usage tracking log file and opening a newly created one for writing subsequent data. The frequency at which rollovers occur is called a *rollover interval*. The default rollover interval is 240 minutes (every 4 hours).

Usage tracking log files that are closed are available for analysis. Setting a lower rollover interval will make usage tracking log files available for analysis sooner, but at the cost of additional overhead.

If the checkpoint interval equals or exceeds the rollover interval, only the rollover occurs explicitly; the checkpoint only occurs implicitly when the old usage tracking log file is closed.

Server Session Management

The Session Manager is used in online mode to monitor activity. The Session Manager shows all users logged into the session, all current query requests for each user, and variables and their values for a selected session. Additionally, the administrator can disconnect any users and kill any query requests with the Session Manager.

How often the Session Manager data refreshes depends on the amount of activity on the system. To refresh the display at any time, click Refresh.

Using the Session Manager

The Session Manager contains an upper and a lower pane:

- The top window, the Session window, shows users currently logged into the Analytics Server. To control the update speed, from the Update Speed drop-down list, choose Normal, High, or Low. Select Pause to keep the display from being refreshed.
- The bottom window contains two tabs.
 - The Request tab shows active query requests for the user selected in the Session window.
 - The Variables tab shows variables and their values for a selected session. You can click the column headers to sort the data.

NOTE: Only 7.7 Analytics Server return information about variables. If the Administration Tool connects to an older-version server online, only the Requests tab will be visible in the Session Manager dialog box.

Table 33 and Table 34 describe the columns in the Session Manager windows.

Table 33. Fields in the Session Window

Column Name	Description
Session ID	The unique internal identifier that the Analytics Server assigns each session when the session is initiated.
User	The name of the user connected.
Client Type	The type of client connected to the server.
Repository	The logical name of the repository to which the session is connected.
Logon Time	The timestamp that shows when the session initially connected to the Analytics Server.
Last Active Time	The timestamp of the last activity on the session.

Table 34. Some Fields in the Request Tab

Column Name	Description
Session ID	The unique internal identifier that the Analytics Server assigns each session when the session is initiated.
Request ID	The unique internal identifier that the Analytics Server assigns each query when the query is initiated.
Start Time	The time of the individual query request.
Last Active Time	The timestamp of the last activity on the query.

To view the variables for a session

- 1 In the Administration Tool, open a repository in online mode and choose Manage > Sessions.
- 2 Select a session and click the Variables tab.
For more information about variables, see [“Using Variables in the Analytics Repository” on page 297](#).
- 3 To refresh the view, click Refresh.
- 4 To close Session Manager, click close.

To disconnect a user from a session

- 1 In the Administration Tool, open a repository in online mode and choose Manage > Sessions.
- 2 Select the user in the Session Manager top window.

3 Click Disconnect.

The user session receives a message indicating that the session was terminated by the administrator. Any currently running queries are immediately terminated, and any outstanding queries to underlying databases are canceled.

4 To close Session Manager, click close.**To kill an active query****1** In the Administration Tool, open a repository in online mode and choose Manage > Sessions.**2** Select the user session that initiated the query in the top window of the Session Manager.

After the user is highlighted, any active query requests from that user are displayed in the bottom window.

3 Select the request you want to kill.**4** Click Kill Request to terminate the highlighted request.

The user receives a message indicating that the query was terminated by the administrator. The query is immediately terminated, and any outstanding queries to underlying databases are canceled.

Repeat this process to kill any other requests.

5 To close Session Manager, click close.

Server Configuration and Tuning

Performance is an extremely important consideration in every decision support system, but it is particularly important in systems that allow queries over the Web. This section describes some important considerations for improving query performance with the Analytics Server.

NQSConfig.INI File Parameters

The NQSConfig.INI file contains configuration and tuning parameters for the Analytics Server. There are parameters to configure disk space for temporary storage, set sort memory buffer sizes, set cache memory buffers, set virtual table page sizes, and a number of other configuration settings that allow you to take full advantage of your hardware's capabilities.

For more information about the NQSConfig.INI file parameters, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Aggregate Tables

You should use aggregate tables to improve query performance. Aggregate tables contain precalculated summarizations of data. It is much faster to retrieve an answer from an aggregate table than to recompute the answer from thousands of rows of detail. The Analytics Server uses aggregate tables automatically, if they have been properly specified in the repository.

For more information and examples of setting up aggregate navigation in a repository, see [“Setting Up Fragmentation Content for Aggregate Navigation” on page 227](#).

Query Caching

Enabling query caching causes the Analytics Server to store query results for reuse by subsequent queries. Caching can dramatically improve the apparent performance of the system for users. For more information about query caching concepts and setup, see [Chapter 12, “Query Caching in the Analytics Server.”](#)

Tune and Index Underlying Databases

The Analytics Server sends queries to databases. For the queries to return in a timely manner, the underlying databases need to be configured, tuned, and indexed correctly. You might need to work with the DBAs of the underlying databases to help identify any problem areas where database tuning is in order.

Different database products will have different tuning considerations. If there are queries that return slowly from the underlying databases, you can capture the SQL of the queries in the query log, then provide them to the DBA for analysis. For more information about configuring query logging on your system, see [“Administering the Query Log” on page 242](#).

12 Query Caching in the Analytics Server

Decision support queries sometimes require large amounts of database processing. The Analytics Server can save the results of a query in cache files and then reuse those results later when a similar query is requested. Using cache, the cost of database processing only needs to be paid once for a query, not every time the query is run.

This section explains query caching and how it is implemented in the Analytics Server.

NOTE: For information about how to use Delivers to seed the Analytics Server Cache, see *Siebel Business Analytics Web Administration Guide*.

This chapter contains the following topics:

- [About the Analytics Server Query Cache on page 257](#)
- [Query Cache Architecture on page 260](#)
- [Configuring Query Caching on page 260](#)
- [Monitoring and Managing the Cache on page 262](#)
- [Purging Cache Programmatically on page 264](#)
- [Strategies for Using the Cache on page 265](#)
- [Cache Event Processing with an Event Polling Table on page 268](#)
- [Making Changes to a Repository on page 273](#)
- [Using the Cache Manager on page 274](#)
- [About the Refresh Interval for XML Data Sources on page 276](#)

About the Analytics Server Query Cache

Administrators can configure the Analytics Server to maintain a local, disk-based cache of query result sets (query cache). The query cache allows the Analytics Server to satisfy many subsequent query requests without having to access back-end databases (such as Oracle or DB2). This reduction in communication costs can dramatically decrease query response time.

As updates occur on the back-end databases, the query cache entries can become stale. Therefore, administrators need to periodically remove entries from the query cache using one of the following methods:

- **Manually.** In the Administration Tool, in the Manage menu, select Cache to open the Cache Manager. Cache Manager provides the maximum flexibility in choosing which cache entries to purge and when to purge them, but it requires direct human involvement. For more information, see [“Using the Cache Manager” on page 274](#).

- **Automatically.** In the Administration Tool, you can disable cache for the system, set caching attributes for a specific physical table, and use Siebel Analytics event tables to purge cache automatically. For additional information about managing cache, see [“Monitoring and Managing the Cache” on page 262](#).
- **Programmatically.** The Analytics Server provides ODBC-extension functions for purging cache entries programmatically. These functions give you the choice and the timing flexibility of Cache Manager with the automation of event tables. You can write your own scripts to call these functions at times that fit your needs. For more information, see [“Purging Cache Programmatically” on page 264](#).

The parameters that control query caching are located in the NQSConfig.INI file described in *Siebel Business Analytics Platform Installation and Configuration Guide*.

NOTE: For information about how to use Delivers to seed the Analytics Server Cache, please see *Siebel Business Analytics Web Administration Guide*.

Advantages of Caching

The fastest way to process a query is to skip the bulk of the processing and use a precomputed answer.

Aggregate tables are examples of precomputed answers. Aggregate tables contain precalculated results for a particular aggregation level. For example, an aggregate table might store sales results for each product by month, when the granularity of detail for the database is at the day level. To create this aggregate table, a process (often a query) computes the results and then stores them in a table in the database.

With query caching, the Analytics Server stores the precomputed results of queries in a local cache. If another query can use those results, all database processing for that query is eliminated. This can result in dramatic improvements in the average query response time.

In addition to improving performance, being able to answer a query from a local cache conserves network resources and processing time on the database server. Network resources are conserved because the intermediate results do not have to come over the network to the Analytics Server. Not running the query on the database frees the database server to do other work. If the database uses a charge back system, it could save money in the budget as well.

Another benefit of using the cache to answer a query is savings in processing time on the Analytics Server, especially if the query results are retrieved from multiple databases. Depending on the query, there might be considerable join and sort processing in the server. If the query is already calculated, this processing is avoided, freeing server resources for other tasks.

To summarize, query caching has the following advantages:

- Dramatic improvement of query performance.
- Less network traffic.
- Reduction in database processing and charge back.
- Reduction in Analytics Server processing overhead.

Initializing Cache Entries for User Ids

To initialize cache entries for user IDs, the Connection Pool needs to be set up for shared login with session variables `VALUEOF(NQ_SESSION.PASSWORD)`, `VALUEOF(NQ_SESSION.USER)` in the login properties. If the shared login is disabled and a user specific database login is used, cache will be shared.

For more information about security, see [Chapter 16, “Security in Siebel Business Analytics.”](#)

Costs of Caching

Query caching has many obvious benefits, but also certain costs:

- Disk space for the cache
- Administrative costs of managing the cache
- Potential for cached results being stale
- Minor CPU and disk I/O on server machine

With proper cache management, the benefits will far outweigh the costs.

Disk Space

The query cache requires dedicated disk space. How much space depends on the query volume, the size of the query result sets, and how much disk space you choose to allocate to the cache. For performance purposes, a disk should be used exclusively for caching, and it should be a high performance, high reliability type of disk system.

Administrative Tasks

There are a few administrative tasks associated with caching. You need to set the cache persistence time for each physical table appropriately, knowing how often data in that table is updated. When the frequency of the update varies, you need to keep track of when changes occur and purge the cache manually when necessary. You can also create a cache event polling table and modify applications to update the polling table when changes to the databases occur, making the system event-driven.

The Analytics Server also provides ODBC-extension functions for purging cache entries programmatically. You can write your own scripts to call these functions at the appropriate times.

Keeping the Cache Up To Date

If the cache entries are not purged when the data in the underlying databases changes, queries can potentially return results that are out of date. You need to evaluate whether this is acceptable. It might be acceptable to allow the cache to contain some stale data. You need to decide what level of stale data is acceptable and then set up (and follow) a set of rules to reflect those levels.

For example, suppose your application analyzes corporate data from a large conglomerate, and you are performing yearly summaries of the different divisions in the company. New data is not going to materially affect your queries because the new data will only affect next year's summaries. In this case, the tradeoffs for deciding whether to purge the cache might favor leaving the entries in the cache.

Suppose, however, that your databases are updated three times a day and you are performing queries on the current day's activities. In this case, you will need to purge the cache much more often, or perhaps consider not using it at all.

Another scenario is that you rebuild your data mart from scratch at periodic intervals (for example, once per week). In this example, you can purge the entire cache as part of the process of rebuilding the data mart, making sure that you never have stale data in the cache.

Whatever your situation, you need to evaluate what is acceptable as far as having noncurrent information returned to the users.

CPU Usage and Disk I/O

Although in most cases it is very minor, query caching does require a small amount of CPU time and adds to the disk I/O. In most cases, the CPU usage is insignificant, but the disk I/O might be noticeable, particularly if queries return large data sets.

Query Cache Architecture

The query cache consists of cache storage space, cache metadata, and cache detection in query compilation.

The process of accessing the cache metadata is very fast. If the metadata shows a cache hit, the bulk of the query processing is eliminated, and the results are immediately returned to the user. The process of adding the new results to the cache is independent of the results being returned to the user; the only effect on the running query is the resources consumed in the process of writing the cached results.

Configuring Query Caching

The query cache is disabled by default. To enable caching, you need to configure the cache storage and decide on a strategy for flushing outdated entries. This section includes information on the tasks necessary to configure the Analytics Server for query caching.

The parameters to control query caching are located in the NQSSConfig.INI file, described in *Siebel Business Analytics Platform Installation and Configuration Guide*.

Configuring the Cache Storage

The following items need to be set up for cache storage in the NQSSConfig.INI file:

- Directories to store the cache files.
- A file to store the cache metadata.

Cache Data Storage Directories

The DATA_STORAGE_PATHS parameter in the CACHE section of the NQSSConfig.INI file specifies one or more directories for query cache storage. These directories are used to store the cached query results and are accessed when a cache hit occurs. For more information, see [“Cache Hits” on page 265](#).

The cache storage directories should reside on high performance storage devices, ideally devoted solely to cache storage. When the cache storage directories begin to fill up, the entries that are least recently used (LRU) are discarded to make space for new entries. The MAX_CACHE_ENTRIES parameter value specifies the maximum number of cache entries allowed in the query cache. The more space you can allocate to the cache, the less often queries will have to access the underlying databases to get the results.

For more information about this configuration parameter, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Metadata Filename

The cache metadata file stores information about queries stored in the cache. The Analytics Server creates and continually updates the metadata file in the location specified in the METADATA_FILE parameter in the CACHE section of the NQSSConfig.INI file. The file should reside on a local, high performance, high reliability storage device. Although not required, it is preferable that the cache metadata file reside on a different disk device than the one specified in DATA_STORAGE_PATHS. Storing the metadata file on a different device from the cache directories eliminates the possibility of I/O bottlenecks between the cache files and the metadata file.

During query compilation, each query is evaluated for potential cache hits. The Analytics Server stores the cache hit information in memory, so this process is very efficient. The metadata file is then accessed for every query that qualifies for a cache hit. The system is designed with a memory buffer for maximum performance and minimum disk I/O. The buffer size is configurable with the BUFFER_POOL_SIZE parameter. Part of the metadata file is stored in this buffer pool. When the buffer pool fills up, it writes the oldest entries to disk and reads in new information from the metadata file. Increasing the value of the BUFFER_POOL_SIZE parameter increases the memory usage and decreases the frequency of reading from and writing to the metadata file.

Maximum Cache Entry Values

You can control the maximum number of rows for any cache entry and the maximum number of cache entries with the MAX_ROWS_PER_CACHE_ENTRY and MAX_CACHE_ENTRIES NQSSConfig.INI file parameters, respectively. Limiting the number of rows is a useful way to avoid using up the cache space with runaway queries that return large numbers of rows. Limiting the total number of cache entries provides another parameter with which to manage your cache storage. If the number of rows a query returns is greater than the value specified in the MAX_ROWS_PER_CACHE_ENTRY parameter, the query is not cached.

For the syntax of these parameters, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Aggregates

Typically, if a query gets a cache hit from a previously executed query, then the new query is not added to the cache. The `POPULATE_AGGREGATE_ROLLUP_HITS` parameter overrides this default when the cache hit occurs by rolling up an aggregate from a previously executed query.

Enabling Query Caching

After configuring the cache storage and deciding on one or more cache management strategies, as discussed in [“Monitoring and Managing the Cache” on page 262](#), you can enable query caching.

To enable query caching

- 1 Set the `ENABLE` parameter in the `CACHE` section of the `NQSConfig.INI` file to `YES`.
- 2 Restart the Analytics server.

Disabling Query Caching

This section explains how to disable query caching.

To disable query caching

- 1 Set the `ENABLE` parameter in the `CACHE` section of the `NQSConfig.INI` file to `NO`.
- 2 Restart the Analytics server.

Monitoring and Managing the Cache

To manage the changes in the underlying databases and to monitor cache entries, you need to develop a *cache management strategy*. You need a process to invalidate cache entries when the data in the underlying tables that compose the cache entry have changed, as well as a process to monitor, identify, and remove any undesirable cache entries.

Choosing a Cache Management Strategy

The choice of a cache management strategy depends on the volatility of the data in the underlying databases and the predictability of the changes that cause this volatility. It also depends on the number and types of queries that comprise your cache, as well as the usage those queries receive. This section provides an overview of the various approaches to cache management.

Disable Caching for the System

You can disable caching for the whole system by setting the `ENABLE` parameter to `NO` in the `NQSConfig.INI` file and restarting the Analytics Server. Disabling caching stops all new cache entries and stops any new queries from using the existing cache. Disabling caching allows you to enable it at a later time without losing any entries already stored in the cache.

Temporarily disabling caching is a useful strategy in situations where you might suspect having stale cache entries but want to verify if they are actually stale before purging those entries or the entire cache. If you find that the data stored in the cache is still relevant, or after you have safely purged problem entries, you can safely enable the cache. If necessary, purge the entire cache or the cache associated with an entire business model before enabling the cache again.

Caching and Cache Persistence Timing for Specified Physical Tables

You can set a cachable attribute for each physical table, allowing you to specify if queries for a that table will be added to the cache to answer future queries. If you enable caching for a table, any query involving the table is added to the cache. All tables are cachable by default, but some tables may not be good candidates to include in the cache unless you use the Cache Persistence Time settings. For example, perhaps you have a table that stores stock ticker data, that is updated every minute. You could use the Cache Persistence Time settings to purge the entries for that table every 59 seconds.

You can also use the Cache persistence time field to specify how long the entries for this table should be kept in the query cache. This is useful for data sources that are updated frequently.

To set the caching attributes for a specific physical table

- 1 In the Physical layer, double-click the physical table.
- 2 In the Physical Table properties dialog box, in the General tab, make one of the following selections:
 - To enable caching, select the Make table cachable check box.
 - To prevent a table from ever being cached, clear the Make table cacheable check box.
- 3 To set an expiration time (maximum lifetime), perform the following steps:
 - a In the Cache persistence time drop-down list, select a value.
If you select Infinite or until you select a different value, the Cache Persistence time field will not be available.
 - b Complete the Cache persistence time field.
- 4 Click OK.

Configure Analytics Server Event Polling Tables

Analytics Server event polling tables store information about updates in the underlying databases. An application (such as one that loads data into a data mart) could be configured to add rows to an event polling table each time a database table is updated. The Analytics Server polls this table at set intervals and invalidates any cache entries corresponding to the updated tables. Event polling tables can be your sole method of cache management or can be used in conjunction with other cache management schemes. Event tables offer less flexibility about choice of cache entries and the timing of purges. For more information about event polling tables, see [“Setting Up Event Polling Tables on the Physical Databases” on page 268](#).

Purging Cache Programmatically

The Analytics Server provides ODBC-extension functions for purging cache entries programmatically. These functions are particularly useful for embedding in an Extract, Transform, and Load (ETL) task. For example, after a nightly ETL is performed, all Analytics Server cache can be purged. If only the fact table was modified, only cache related to that table can be purged. In some cases, you might need to purge the cache entries associated with a specific database.

NOTE: Only administrators have the right to purge cache. Therefore scripts that call these ODBC-extension functions must run under an administrator logon ID.

The following ODBC functions affect cache entries associated with the repository specified by the ODBC connection:

- **SAPurgeCacheByQuery.** Purges a cache entry that exactly matches a specified query. For example, using the following query, you would have a query cache entry that retrieves the names of all employees earning more than \$100,000:

```
select lastname, firstname from employee where salary > 100000;
```

The following call programmatically purges the cache entry associated with this query:

```
Call SAPurgeCacheByQuery('select lastname, firstname from employee where salary > 100000');
```

- **SAPurgeCacheByTable.** Purges all cache entries associated with a specified physical table name (fully qualified) for the repository to which the client has connected.

This function takes up to four parameters representing the four components (database, catalog, schema and table name proper) of a fully qualified physical table name. For example, you might have a table with the fully qualified name of DBName.CatName.SchName.TabName. To purge the cache entries associated with this table in the physical layer of the analytics repository, execute the following call in a script:

```
Call SAPurgeCacheByTable('DBName', 'CatName', 'SchName', 'TabName');
```

NOTE: Wild cards are not supported by the Analytics Server for this function. Additionally, DBName and TabName cannot be null. If either one is null, you will receive an error message.

- **SAPurgeAllCache.** Purges all cache entries. The following is an example of this call:

```
Call SAPurgeAllCache();
```

- **SAPurgeCacheByDatabase.** Purges all cache entries associated with a specific physical database name. A record is returned as a result of calling any of the ODBC procedures to purge the cache. This function takes one parameter that represents the physical database name and the parameter cannot be null. The following is an example of this call:

```
Call SAPurgeCacheByDatabase('DBName');
```


About Result Records

The result record contains two columns. The first column is a result code and the second column is a short message describing result of the purge operation. The following list contains examples of result records:

Result Code	Result Message
1	SAPurgeCacheByDatabase returns successfully.
E_Execution_CacheNotEnabled	Operation not performed because caching is not enabled.
E_Execution_NonExistingDatabase	The database specified does not exist.

Strategies for Using the Cache

One of the main advantages of query caching is to improve apparent query performance. It may be valuable to *seed* the cache during off hours by running queries and caching their results. A good seeding strategy requires that you know when cache hits occur.

If you want to seed the cache for all users, you might seed the cache with the following query:

```
Select User, SRs
```

After seeding the cache using `Select User, SRs`, the following queries will all be cache hits:

```
Select User, SRs where user = valueof(nq_SESSION.USER) (and the user was USER1)
```

```
Select User, SRs where user = valueof(nq_SESSION.USER) (and the user was USER2)
```

```
Select User, SRs where user = valueof(nq_SESSION.USER) (and the user was USER3)
```

Cache Hits

When caching is enabled, each query is evaluated to see if it qualifies for a cache hit. A cache hit means that the server was able to use cache to answer the query and did not go to the database at all.

NOTE: The Analytics Server can use query cache to answer queries at the same or higher level of aggregation.

A cache hit occurs only if all of the conditions described in this section are met.

- **WHERE clause semantically the same or a logical subset.** For the query to qualify as a cache hit, the WHERE clause constraints need to be either equivalent to the cached results, or a subset of the cached results.

A WHERE clause that is a logical subset of a cached query qualifies for a cache hit if the subset meets one of the following criterion:

- A subset of IN list values.

Queries requesting fewer elements of an IN list cached query qualify for a cache hit. For example, the following query:

```
select empl oyeename, regi on
from empl oyeer, geograph y
where regi on i n ( ' EAST' , ' WEST' )
```

qualifies as a hit on the following cached query:

```
select empl oyeename, regi on
from empl oyeer, geograph y
where regi on i n ( ' NORTH' , ' SOUTH' , ' EAST' , ' WEST' )
```

- It contains fewer (but identical) OR constraints than the cached result.
- It contains a logical subset of a literal comparison.

For example, the following predicate:

```
where revenue < 1000
```

qualifies as a cache hit on a comparable query with the predicate:

```
where revenue < 5000
```

- There is no WHERE clause.

If a query with no WHERE clause is cached, *queries that satisfy all other cache hit rules* qualify as cache hits regardless of their WHERE clause.

- **A subset of columns in the SELECT list.** All of the columns in the SELECT list of a new query have to exist in the cached query in order to qualify for a cache hit, or they must be able to be calculated from the columns in the query.
- **Columns in the SELECT list can be composed of expressions on the columns of the cached queries.** The Analytics Server can calculate expressions on cached results to answer the new query, but all the columns have to be in the cached result.

For example, the query:

```
select product, month, averageprice from sales where year = 2000
```

will hit cache on the query:

```
select product, month, dollars, uni tsales from sales where year = 2000
```

because averageprice can be computed from dollars and unitsales (averageprice = dollars/unitsales).

- **Equivalent join conditions.** The resultant joined logical table of a new query request has to be the same as (or a subset of) the cached results to qualify for a cache hit.
- **DISTINCT attribute the same.** If a cached query eliminates duplicate records with DISTINCT processing (for example, SELECT DISTINCT...), requests for the cached columns have to also include the DISTINCT processing; a request for the same column without the DISTINCT processing will be a cache miss.
- **Compatible aggregation levels.** Queries that request an aggregated level of information can use cached results at a lower level of aggregation.

For example, the following query:

```
select supplier, region, city, qtysold
from suppliercity
```

requests the quantity sold at the supplier and region and city level, while the following query:

```
select city, qtysold
from suppliercity
```

requests the quantity sold at the city level. The second query would result in a cache hit on the first query.

- **Limited additional aggregation.** For example, if a query with the column *qtysold* is cached, a request for *RANK(qtysold)* results in a cache miss. Additionally, a query requesting *qtysold* at the country level can get a cache hit from a query requesting *qtysold* at the country, region level.
- **ORDER BY clause made up of columns in the select list.** Queries that order by columns not contained in the select list result in cache misses.

Running a Suite of Queries to Populate the Cache

To maximize potential cache hits, one strategy is to run a suite of queries just for the purpose of populating the cache. The following are some recommendations for the types of queries to use when creating a suite of queries with which to seed the cache.

- Common prebuilt queries.

Queries that are commonly run, particularly ones that are expensive to process, are excellent cache seeding queries. Queries whose results are embedded in Intelligence Dashboards would be good examples of common queries.

- SELECT lists with no expressions.

Eliminating expressions on SELECT list columns expands the possibility for cache hits. A cached column with an expression can only answer a new query with the same expression; a cached column with no expressions can answer a request for that column with any expression. For example, a cached request such as:

```
SELECT QUANTITY, REVENUE...
```

can answer a new query such as:

```
SELECT QUANTITY/REVENUE...
```

but not the reverse.

- No WHERE clause.

If there is no WHERE clause in a cached result, it can be used to answer queries satisfying the cache hit rules for the select list with any WHERE clause that includes columns in the projection list.

In general, the best queries to seed cache with are queries that heavily consume database processing resources and that are likely to be reissued. Be careful not to seed the cache with simple queries that return many rows. These queries (for example, `SELECT * FROM PRODUCTS`, where `PRODUCTS` maps directly to a single database table) require very little database processing. Their expense is network and disk overhead—factors that caching will not alleviate.

NOTE: When the Analytics Server refreshes repository variables, it will examine business models to determine if they reference those repository variables. If they do, the Analytics Server purges all cache for those business models.

Cache Event Processing with an Event Polling Table

The use of a Analytics Server event polling table (event table) is a way to notify the Analytics Server that one or more physical tables have been updated. Each row that is added to an event table describes a single update event, such as an update occurring to the `Product` table in the `11308Production` database. The Analytics Server cache system reads rows from, or *polls*, the event table, extracts the physical table information from the rows, and purges stale cache entries that reference those physical tables.

The event table is a physical table that resides on a database accessible to the Analytics Server. Regardless of where it resides—in its own database, or in a database with other tables—it requires a fixed schema, described in [Table 35 on page 269](#). It is normally exposed only in the Physical layer of the Administration Tool, where it is identified in the Physical Table dialog box as being a Analytics Server event table.

The use of event tables is one of the most accurate ways of invalidating stale cache entries, and it is probably the most reliable method. It does, however, require the event table to be populated each time a database table is updated (see [“Populating the Analytics Server Event Polling Table” on page 272](#)). Also, because there is a polling interval in which the cache is not completely up to date, there is always the potential for stale data in the cache.

A typical method of updating the event table is to include SQL `INSERT` statements in the extraction and load scripts or programs that populate the databases. The `INSERT` statements add one row to the event table each time a physical table is modified. After this process is in place and the event table is configured in the analytics repository, cache invalidation occurs automatically. As long as the scripts that update the event table are accurately recording changes to the tables, stale cache entries are purged automatically at the specified polling intervals.

Setting Up Event Polling Tables on the Physical Databases

This section describes how to set up the Analytics Server event polling tables on physical databases.

Polling Table Structure

You can set up a physical event polling table on each physical database to monitor changes in the database. You can also set up the event table in its own database. The event table should be updated every time a table in the database changes. The event table needs to have the structure shown in [Table 35](#); some columns can contain null values depending on where the event table resides.

The column names for the event table are suggested; you can use any names you want. However, the order of the columns has to be the same as shown in [Table 35](#). Sample CREATE TABLE statements to create an event polling table are shown in [“Sample Event Polling Table CREATE TABLE Statements” on page 270](#).

Table 35. Event Polling Table Column Names

Event Table Column Name	Data Type	Description
UpdateType	INTEGER	Specify a value of 1 in the update script to indicate a standard update. (Other values are reserved for future use.) Values cannot be null.
UpdateTime	DATETIME	The time when the update to the event table occurs. This needs to be a key (unique) value that increases for each row added to the event table. To make sure a unique and increasing value, specify the current timestamp as a default value for the column. For example, specify <code>DEFAULT CURRENT_TIMESTAMP</code> for Oracle 8i. Values cannot be null.
DatabaseName	CHAR or VARCHAR	The name of the database where the physical table that was updated resides. This is the name of the database as it is defined in the Physical layer of the Administration Tool. For example, if the physical database name is 11308Production, and the database name that represents it in the Administration Tool is SQL_Production, the polled rows in the event table has to contain SQL_Production as the database name. Populate the DatabaseName column only if the event table does not reside in the same database as the physical tables that were updated. Otherwise, set it to the null value.
CatalogName	CHAR or VARCHAR	The name of the catalog where the physical table that was updated resides. Populate the CatalogName column only if the event table does not reside in the same database as the physical tables that were updated. Otherwise, set it to the null value.

Table 35. Event Polling Table Column Names

Event Table Column Name	Data Type	Description
SchemaName	CHAR or VARCHAR	The name of the schema where the physical table that was updated resides. Populate the SchemaName column only if the event table does not reside in the same database as the physical tables being updated. Otherwise, set it to the null value.
TableName	CHAR or VARCHAR	The name of the physical table that was updated. The name has to match the name defined for the table in the Physical layer of the Administration Tool. Values cannot be null.
Other	CHAR or VARCHAR	Reserved for future enhancements. This column must be set to a null value.

The Analytics Server needs to have read and write permission on the event polling table. The server reads the event table at specified intervals to look for changed data. Applications add rows to the event table when database tables are modified (for example, during a load operation). When there are rows in the event table, there is changed data in the underlying databases. The server then invalidates any cache entries corresponding to the changed physical tables and periodically deletes obsolete rows from the event table. The next time it checks the event table, the process repeats.

NOTE: A single event polling table cannot be shared by multiple Analytics Servers. When you set up multiple Analytics Servers, you need to create an event polling table for each one.

To allow Analytics Server to have write access to the event polling table but not to any other tables in a database, perform the following tasks:

- Create a separate physical database in the Physical layer of the Administration Tool with a privileged connection pool.
- Assign a user to the connection pool that has delete privileges.
- Populate the privileged database with the event table.

The Analytics Server will have write access to the event polling table, but not to any tables that are used to answer user queries.

Sample Event Polling Table CREATE TABLE Statements

The following are sample CREATE TABLE statements for SQL Server 7.0 and Oracle 8i. These CREATE TABLE statements create the structure required for a Analytics Server event polling table. In these statements, the table created is named UET. It resides in the same database as the physical tables that are being updated.

NOTE: The column lengths need to be large enough to represent the object names in your repository.

The following is the CREATE TABLE statement for SQL Server 7.0:

```
// SQL Server 7.0 Syntax
create table UET (
    UpdateType Integer not null,
    UpdateTime datetime not null DEFAULT CURRENT_TIMESTAMP,
    DBName      char(40) null,
    CatalogName varchar(40) null,
    SchemaName  varchar(40) null,
    TableName   varchar(40) not null,
    Other       varchar(80) null DEFAULT NULL
)
```

The following is the CREATE TABLE statement for Oracle 8i:

```
// Oracle 8i syntax
create table UET (
    UpdateType Integer not null,
    UpdateTime date DEFAULT SYSDATE not null,
    DBName      char(40) null,
    CatalogName varchar(40) null,
    SchemaName  varchar(40) null,
    TableName   varchar(40) not null,
    Other       varchar(80) DEFAULT NULL
);
```

You might need to modify these CREATE TABLE statements slightly for different versions of SQL Server and Oracle, or for other databases. Additionally, if you want to specify any explicit storage clauses, you need to add the appropriate clauses to the statements.

Making the Event Polling Table Active

After the table is created on the physical database, you can make it active in the Analytics Server.

To make the polling table active

- 1 Import the table to the Physical layer.
- 2 Include it in the group of Analytics Server event polling tables using the Tools > Utilities > Siebel Event Tables menu item, and set a polling interval.

To import the polling table into the Physical layer, perform the following steps from an open repository.

To import the table into the Physical layer

- 1 Select File > Import...
- 2 Select the data source containing the event table to import, and then click OK.
The Import dialog box appears.
- 3 Check the Tables option to import the table metadata.

- 4 Navigate to the event polling table, select the table and either click the Import button or drag and drop it into the Physical layer.

This imports the event table to the Physical layer. If you have multiple event polling tables, repeat this procedure for each event table.

NOTE: Be sure the data source specified for the event table has read and write access to the event table. The repository will both read the table and delete rows from it, so it needs write permission. Event tables do not need to be exposed in the Business Model and Mapping layer.

After one or more polling tables are present in the Physical layer, you need to include them in the group of event polling tables.

To mark the table object as an Event Polling Table

- 1 Click on the Tools > Utilities menu item.
- 2 Select the option Siebel Event Tables from the list of options.
- 3 Click Execute.
- 4 Select the table to register as an Event Table and click the >> button.
- 5 Specify the polling frequency in minutes, and click OK.

The default value is 60 minutes.

NOTE: You should not set the polling frequency to less than 10 minutes. If you want a very short polling interval, consider marking some or all of the tables noncacheable.

When a table has been registered as a Analytics Server event table, the table properties change. Registration as an event table removes the option to make the table cacheable, as there is no reason to cache results from an event polling table.

Populating the Analytics Server Event Polling Table

The Analytics Server does not populate the event polling table. The event table is populated by inserting rows into it each time a table is updated. This process is normally set up by the database administrator; typically, the load process is modified to insert a row into the polling table each time a table is modified. This can be done from the load script, using database triggers (in databases that support triggers), from an application, or manually. If the process of populating the event table is not done correctly, the Analytics Server cache purging will be affected; the server assumes the information in the polling table to be correct and up to date.

Troubleshooting Problems with an Event Polling Table

If you experience problems with cache polling, you can search the Analytics Server activity logs for any entries regarding the server's interaction with the event table.

- The NQServer.log file logs activity automatically about the Analytics Server. The default location for this file is the Log folder in the Analytics Server software installation folder (\Siebel Analytics\Log). Log entries are self-explanatory and can be viewed using a text editor.
- When the Analytics Server polls the event table, it will log the queries in the NQQuery.log file using the administrator user ID unless the logging level for the administrator is set to 0. You should set the logging level to 2 for the administrator user ID to provide the most useful level of information. The default location for the NQQuery.log file is the Log folder in the Analytics Server software installation folder (\Siebel Analytics). For more information about user-level logging, see [“Query Caching in the Analytics Server” on page 257](#).

Making Changes to a Repository

When you modify analytics repositories, the changes can have implications for entries that are stored in the cache. For example, if you change the definition of a physical object or a dynamic repository variable, cache entries that reference that object or variable may no longer be valid. These changes might result in the need to purge the cache. There are three scenarios to be aware of—when the changes occur in online mode, when they occur in offline mode, and when you are switching between repositories.

Online Mode

When you modify an analytics repository in online mode, any changes you make that will affect cache entries automatically result in a purge of all cache entries that reference the changed objects. The purge occurs when you check in the changes. For example, if you delete a physical table from a repository, all cache entries that reference that table are purged upon check in. Any changes made to a business model in the Business Model and Mapping layer will purge all cache entries for that business model.

Offline Mode

When you modify an analytics repository in offline mode, you might make changes that affect queries stored in the cache and render those cached results obsolete. Because the repository is not loaded by the server during offline mode edits, the server has no way of determining if the changes made affect any cached entries. The server therefore does *not* automatically purge the cache after offline changes. If you do not purge the cache, there might be invalid entries when the repository is next loaded. Unless you are sure that there are no entries in the cache that are affected by your offline changes, you should purge the cache for any business model you have modified.

Switching Between Repositories

If you intend to remove a repository from the Analytic Server’s configuration, make sure to purge the cache of all cache entries that reference the repository. Failure to do so will result in a corrupted cache. For information, see [“Purging Cache” on page 275](#).

Using the Cache Manager

The Cache Manager provides administrators the capability of viewing information about the entire query cache, as well as information about individual entries in the query cache associated with the open repository. It also provides the ability to select specific cache entries and perform various operations on those entries, such as viewing and saving the cached SQL call, or purging them.

To open the Cache Manager

■ In the Administration Tool toolbar, select Manage > Cache.

Select the Cache tab on the left explorer pane to view the cache entries for the current repository, business models, and users. The associated cache entries are reflected in the right pane, with the total number of entries shown in the view-only field at the top.

The cache entry information and its display sequence is controlled by your Options settings (select Edit > Options... from the Cache Manager, or Tools > Options > Cache Manager tab from the Administration Tool). Information may include the options in [Table 36](#).

Table 36. Cache Options

Option	Description
Query Server	The Analytics Server that serviced the query.
User	The ID of the user who submitted the query that resulted in the cache entry.
Created	The time the cache entry's result set was created.
Last used	The last time the cache entry's result set satisfied a query. (After an unexpected shutdown of the Analytics Server, the last used time may temporarily have a <i>stale</i> value—a value that is older than the true value.)
Creation elapsed time	The time, in seconds, needed to create the result set for this cache entry. NOTE: The value stored in the cache object descriptor on disk is in units of milliseconds. The value is converted to seconds for display purposes.
Row count	The number of rows generated by the query.
Row size	The size of each row (in bytes) in this cache entry's result set.
Full size	Full size is the maximum size used, considering variable length columns, compression algorithm, and other factors. The actual size of the result set will be smaller than Full size.
Column count	The number of columns in each row of this cache entry's result set.
SQL	The SQL statement associated with this cache entry.
Use count	The number of times this cache entry's result set has satisfied a query (since Analytics Server startup).
Business model	The name of the business model associated with the cache entry.

Expand the repository tree to display all the business models with cache entries, and expand the business models to display all users with cache entries. The right pane displays only the cache entries associated with the selected item in the hierarchical tree.

Displaying Global Cache Information

Select Action > Show Info... to display global cache information. [Table 37](#) describes the information that appears in the Global Cache Information window.

Table 37. Global Cache Information

Column	Description
Number of entries currently in cache	The current number of entries in your global cache. These entries may relate to multiple repositories.
Maximum allowable number of entries in cache	The maximum number of entries that may be in your cache, from the MAX_CACHE_ENTRIES parameter in the NQSSConfig.INI file.
Amount of space still available for cache storage use	The amount of space, in megabytes, still available for cache storage.
Amount of space used on disks containing cache related files	The <i>total</i> amount of space, in megabytes, used on the disk containing cache-related files (not just space used for the cache-related files).
Maximum allowable number of rows per cache entry result set	The maximum number of rows allowed for each cache entry's result set, from the MAX_ROWS_PER_CACHE_ENTRY parameter in the NQSSConfig.INI file.
Number of queries satisfied from cache since startup of Analytics Server	Cache hits, since the last time the Analytics Server was started.
Number of queries not satisfied from cache since startup of Analytics Server	Cache misses, since the last time the Analytics Server was started.

With the Cache Manager as the active window, press F5, or select Action > Refresh to refresh the display. This retrieves the current cache entries for the repository you have open, as well as the current global cache information. If the DSN is clustered, information about all repositories in the cluster will be displayed.

Purging Cache

Purging cache is the process of deleting entries from the query cache. You can purge cache entries in the following ways:

- Manually, using the Administration Tool Cache Manager facility (in online mode).
- Automatically, by setting the Cache Persistence Time field in the Physical Table dialog box for a particular table.
- Automatically, by setting up a Analytics Server event polling table.
- Automatically, as the cache storage space fills up.

To purge the cache manually with the Cache Manager facility

- 1 Use the Administration Tool to open a repository in online mode.
- 2 Select Manage > Cache to open the Cache Manager dialog box.
- 3 Select Cache or Physical mode by selecting the appropriate tab in the left pane.
- 4 Navigate the explorer tree to display the associated cache entries in the right pane.
- 5 Select the cache entries to purge, and then select Edit > Purge to remove them.
 - In Cache mode, select the entries to purge from those displayed in the right pane.
 - In Physical mode, select the database, catalog, schema or tables to purge from the explorer tree in the left pane.

In Cache mode, you can purge:

- One or more selected cache entries associated with the open repository.
- One or more selected cache entries associated with a specified business model.
- One or more selected cache entries associated with a specified user within a business model.

In Physical mode, you can purge:

- All cache entries for all tables associated with one or more selected databases.
- All cache entries for all tables associated with one or more selected catalogs.
- All cache entries for all tables associated with one or more selected schemas.
- All cache entries associated with one or more selected tables.

Purging deletes the selected cache entries and associated metadata. Select Action > Refresh or press F5 to refresh your cache display.

About the Refresh Interval for XML Data Sources

This section provides information about the refresh interval for XML data sources.

For more information, see [“Using XML as a Data Source for Siebel Business Analytics” on page 347](#).

Typically, XML data sources are updated frequently and in real time. Setting a refresh interval for XML data sources is analogous to setting cache persistence for database tables. The refresh interval is a time interval after which the XML data sources are to be queried again directly, rather than using results in cache. This refresh interval is specified on the XML tab of the Connection Pool dialog box.

The default interval setting is Infinite, meaning that the XML data source is not automatically refreshed.

The refresh interval setting specifies the time interval after which the Analytics Server XML Gateway connection will be refreshed.

- For URLs that begin with `http://` or `https://`, the gateway will refresh when it detects that the interval has expired.
- For URLs that reside on a local or network drive, the gateway will refresh when the interval has expired and the system detects that the URLs have been modified.

13 Connectivity and Third-Party Tools

The Analytics Server provides the functionality for you to connect to it through many client tools and applications. This section contains the following topics:

- [Configuring Analytics ODBC Data Source Names \(DSNs\) on page 279](#)
- [Third-Party Tools and Relational Data Source Adapters on page 281](#)
- [Importing Metadata on page 281](#)
- [Using IBM DB2 Cube Views with Siebel Business Analytics on page 282](#)
- [ODBC Conformance Level on page 295](#)

Configuring Analytics ODBC Data Source Names (DSNs)

In a non-English environment, you cannot use a direct ODBC connection to Analytics Server. Only the Siebel Business Analytics Web client can connect directly to Analytics Server in a non-English environment.

This procedure applies to Windows-based operating systems.

To create a new data source

- 1 Open the Windows ODBC Control Panel applet by selecting Start > Settings > Control Panel, and then double-click the ODBC Data Sources icon.

If you are running Windows 2000 or XP, the Data Sources (ODBC) icon is available as an Administrative Option.
- 2 In the ODBC Data Source Administrator dialog box, click the System DSN tab, and then click Add.
- 3 Select the Siebel Analytics driver from the Create New Data Source dialog, and then click Finish.
The first DSN Configuration screen appears.
- 4 Type a name for the data source in the Name field.
- 5 (Optional) Type a description in the Description field.
- 6 If this data source will not participate in a cluster, in the Server field at the bottom of the screen, select the machine on which the Analytics Server is running.

If the server name does not appear in the drop-down list, type the name in the Server field. This needs to be the NetBIOS name (computer name) of the machine.
- 7 If this data source is to participate in a cluster, do the following:

- a** Select the option Is this a Clustered DSN?

This causes the fields Primary Controller and Secondary Controller to become active, and the Server field to become inactive.

- b** Type the name of the machine that is specified as the primary Cluster Controller (from the parameter PRIMARY_CONTROLLER in the NQClusterConfig.INI file). This needs to be the NetBIOS name (computer name) of the machine.
- c** If a secondary Cluster Controller has been specified (from the parameter SECONDARY_CONTROLLER in the NQClusterConfig.INI file), type the name of the machine in the Secondary Controller field. The computer name must be unique from that of the primary Cluster Controller.
- d** To test the connection to the Cluster Controller, click Test Connect.

A message indicates if the connection was tested successfully. If the test is not successful, correct any error identified in the message and test the connection again.

- 8** In the next DSN Configuration screen, type a valid user ID and password for the repository to which you want the data source to connect. If you are using Windows operating system authentication, leave this field blank. You will then log into the Analytics Server with the logon ID of your Windows account.
- 9** If you want to save your logon ID in the repository, check the option Save login ID.
If you check this option, you will not have to type your logon information each time you connect.
- 10** In the Port field, specify the TCP/IP port the Analytics Server is using for client/server communications.

The default port is 9703. This port number should match the port number specified in the parameter RPC_SERVICE_OR_PORT in the Server section in the NQSCConfig.INI file. If you change the port number in the configuration file, remember to reconfigure any affected ODBC data sources to use the new port number.

NOTE: The default client/server communication method for the Analytics Server has changed from Distributed component object model (DCOM) to TCP/IP. Support for DCOM will be discontinued in a future release. For sites already running the Analytics Server that want to continue to use DCOM until support is discontinued, leave this field set to its default value and define a Windows system environment variable named NQUIRE_DCOM to force the usage of DCOM. Set the variable value to 1. (To define a system environment variable, select System from the Control Panel, click the Advanced tab, and then click the Environment Variables button to open the Environment Variables dialog box.)

- 11** If you want to connect to a repository other than the default repository, select the option Change the default repository to, and then type the logical name of the repository (as it appears in the NQSCConfig.INI file) to which you want to connect in the field below the check box.

If this option is not selected, the data source will connect to the repository marked as the default repository in the NQSCConfig.INI file, or to the first repository listed if none of the entries is marked as the default.

- 12** Select the option Connect to Analytics Server to obtain default settings for additional configuration.

The setup will attempt to connect to the server to obtain information about the business models in the repository. If you do not select this option, you can still configure the DSN by manually entering the information in the next configuration screen.

- 13** Click Next to advance to the next window.

- 14** To change the default catalog, select the option Change the default catalog to, and then type the name of the catalog in the field below the check box.

The default catalog is the catalog folder that appears at the top of the Presentation layer in the Administration Tool. For the DSN used by Siebel Business Analytics Web, it is better to leave this check box clear with the drop-down box showing no entry.

You can also specify user IDs and passwords for the underlying databases to which the Analytics Server connects. If you specify database user IDs and passwords, those are used to connect to the databases if user-specific database logon information is configured in the connection pools, as described in [“Creating or Changing Connection Pools” on page 68](#). The database-specific user IDs and passwords allow privileged users to connect to the underlying databases at the level of authority granted to those users in the databases.

- 15** At this point, you can change the password for the Siebel Business Analytics user the DSN logs in as (if the server is running in a writable mode). To change the password, you must have entered your logon information and selected the option Connect to Siebel Analytics in the previous screen. The new password is stored in encrypted form in the repository.

Third-Party Tools and Relational Data Source Adapters

The Analytics Server allows connectivity between a wide variety of client tools and a wide variety of data sources. For information, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

Importing Metadata

You can import metadata from a data source to an analytics repository. The metadata is used to establish physical table information in the Physical layer of the Administration Tool.

Metadata imports to an analytics repository have to occur through an ODBC connection to the underlying data source; native database gateways for metadata import are only supported for DB2 (using DB2 CLI) and XML. Metadata can also be imported from a text file. Metadata cannot be imported for multidimensional data sources.

For the metadata import procedure, see [“Process of Creating the Physical Layer from Relational Data Sources” on page 58](#).

Using Query and Reporting Tools

You can connect to the Analytics Server with a wide variety of ODBC-compliant query and reporting tools. Connecting with a query tool is a matter of configuring a data source using the analytics ODBC driver and then using the analytics DSN to connect to a repository from the query tool.

The Presentation layer allows you to configure the presentation of a business model to be consistent with the rules and conventions of your tools to take advantage of the Analytics Server's analytical engine and data abstraction. This makes it much easier to include columns involving complex aggregation and calculation rules in queries and reports. Also, if your organization is currently using query and reporting tools, using the Analytics Server as a data source will make these tools more valuable and will simplify the work entailed when using them.

Using IBM DB2 Cube Views with Siebel Business Analytics

The term IBM DB2 Cube Views is a registered trademark of IBM.

This section contains the following topics:

- [About Using IBM DB2 Cube Views with Siebel Business Analytics on page 282](#)
- [Generating the Import Files on page 283](#)
- [Process of Deploying Cube Metadata on page 292](#)

About Using IBM DB2 Cube Views with Siebel Business Analytics

The term IBM DB2 Cube Views is a registered trademark of IBM. For information about platform compatibility, see *System Requirements and Supported Platforms*.

This feature enhances the data warehouse performance and functionality of a database. It allows the DB2 database to store metadata about the logical relationships of the data residing in the database. Additionally, it accelerates data warehouse queries by using more efficient DB2 materialized query tables (MQTs). These MQTs preaggregate the relational data and improve query performance.

When processing queries, the DB2 Query Rewrite functionality routes queries to the MQTs when possible. Because these tables are smaller than the underlying base tables and the data has been preaggregated, the queries that are rerouted to them might run faster.

Siebel Analytics CubeViews Generator works as a metadata bridge to convert the Siebel Business Analytics proprietary metadata into a IBM Cube Views XML file. After converting metadata into an XML file, you use IBM Cube Views to import the translated metadata into your DB2 database and store it in IBM Cube Views metadata catalog tables. After importing the metadata, you use the IBM Optimization Advisor to generate scripts to create materialized query tables (MQT) and their indexes. The deployed MQTs are used by the DB2 Query Reroute Engine to optimize incoming application queries.

NOTE: DB2 provides an API (implemented as a stored procedure) that passes XML documents as arguments to create, modify, delete, or read the metadata objects. For more information about IBM Cube Views, see your IBM DB2 documentation.

For additional information about using IBM Cube Views with Siebel Business Analytics, see the following topics:

- [Generating the Import Files on page 283](#)
- [Process of Deploying Cube Metadata on page 292](#)

Generating the Import Files

The Siebel Analytics CubeViews Generator creates the files needed to import analytics metadata into a DB2 database.

This section contains the following topics:

- [Running the CubeViews Generator on page 283](#)
- [About the Cube Metadata Input File on page 284](#)
- [About the CubeViews Generator Output Files on page 285](#)
- [Troubleshooting CubeViews Generator Errors on page 285](#)
- [About Mapping Metadata Objects on page 286](#)
- [Metadata Conversion Rules and Error Messages on page 287](#)

Running the CubeViews Generator

The CubeViews Generator is invoked from the command line or embedded in a batch file. The command-line executable is named SACubeViewsGen.exe, and has the following syntax:

```
-r "PathAndRepositoryFileName" -u <UserName> -p <Password> -f "InputFileNameAndPath"
```

Table 38 on page 284 contains descriptions of the parameters in the command-line executable file.

Table 38. Parameters in the SACubeViewsGen.exe

Parameter	Definition	Additional Information
-r	Repository file name and full path	Quotation marks are required for the filename and path only if the file path is in long format or has spaces. Use the full path if the file is not in the current directory.
-u	User name	User name that will allow access to the repository.
-p	Password	Password for the user name. If the repository password is empty, do not use the password parameter.
-f	Input file name and full path	Quotation marks are required for the filename and path only if the file path is in long format or has spaces. Use the full path if the file is not in the current directory. You specify input files so that you do not have to type all the required information at the command line, and so that you can type international characters. For more information about the input file, see “About the Cube Metadata Input File” on page 284 .

About the Cube Metadata Input File

The input file is a text file containing the parameters in the following 4 lines:

BUSINESS_MODEL = "*[name of business model]*"

PHYSICAL_DATABASE= "*[name of physical database]*"

RUN_AS_USER="*[username]*"

OUTPUT_FOLDER="*[full path and filename]*"

Table 39 on page 284 contains a description of the cube metadata input file parameters.

Table 39. Cube Metadata Input File Parameters

Input File Name	Description
BUSINESS_MODEL	The name of the business model in the logical layer of the analytics repository that contains the metadata that you want to export to the DB2 database. If the business model is not found in the repository, an error appears.
PHYSICAL_DATABASE	The name of the database in the physical layer of the analytics repository that contains the metadata that you want to export to the DB2 database. When the business model derives from more than one database, then it eliminates metadata from all databases other than the one specified here. When the physical database is not found in the repository, an error appears.

Table 39. Cube Metadata Input File Parameters

Input File Name	Description
RUN_AS_USER	The username of the user whose visibility needs to be duplicated for the metadata export. This cannot be empty. If the user is not found in the repository, an error appears.
OUTPUT_FOLDER	The full path and filename of the folder to which the XML files and the alias-SQL file will be written. If the folder does not exist when you run the CubeViews Generator, it will be created. For more information about the output files, see “About the CubeViews Generator Output Files” on page 285 .

About the CubeViews Generator Output Files

Running the CubeViews Generator creates the following files in the specified output folder:

- **XML files (encoded in UTF8).** One XML file is created for each specified business model. It contains all objects that were converted to cubes. Additionally, objects in the repository will be mapped to similar objects in the IBM Cube Views metadata. For a list of objects that will not be converted, see [“Metadata Conversion Rules and Error Messages” on page 287](#).

The name of the XML file will match the business model name (without spaces), followed by the XML extension. For example, SalesResults.xml. The following is the syntax of the XML file name:

```
[BusinessModel NameWithNoSpaces].xml
```

- **A SQL file containing the alias generation DLL.** A SQL file is created for each specified business model only if aliases exist in the physical layer databases that are referenced in the specified business model. The alias file contains SQL commands that will create the aliases in the DB2 database. The name of the SQL file will match the business model name (without spaces), followed by the SQL extension. For example, SalesResults-alias.sql. The following is the syntax of the alias-SQL file name:

```
[BusinessModel NameWithNoSpaces]-alias.sql
```

Troubleshooting CubeViews Generator Errors

CubeViews Generator errors indicate that the CubeViews Generator was unable to complete some or all of its tasks. After starting the CubeViews Generator, you might see the following errors on your screen:

- Unable to write to Log file : @1%ls.
The log file specified in the NQSCfg.INI file might contain the wrong path, the user might not have write permissions to that folder, or the disk could be out-of-space.
- Run_as_user, @1%ls, is invalid.
The user name is incorrect.

- Repository, @1%ls, is invalid or corrupt.
The repository name might be incorrect, it might not exist in the given path, or the user might not have permission to read it.
- Physical Database, @1%ls, is invalid.
The physical database name is incorrect.
- Business Model, @1%ls, is invalid.
The business model name is correct.
- Authentication information provided is invalid.
The specified username or password is incorrect.
- Path : "@1%ls" is invalid.
The path or file name is incorrect.

About Mapping Metadata Objects

When the CubeViews generator creates the XML file, it also maps the metadata objects in the analytics repository to similar objects in the IBM Cube Views metadata. For a list of conversion rules used during this mapping, see [“Metadata Conversion Rules and Error Messages” on page 287](#).

The following is a list of IBM Cube Views metadata objects:

- Attribute
- Join
- Measure
- Fact
- Dimension
- Hierarchy
- Level
- Cube Model
- Cube, Cube Fact, Cube Dimension, Cube Hierarchy, and Cube Level

[Table 40 on page 286](#) contains a list of Siebel Business Analytics metadata objects.

Table 40. Siebel Business Analytics Metadata Objects

Metadata	Description
Logical Fact Table	Made up of one or more logical fact table sources.
Logical Fact Table Source	Made up of joins between multiple physical fact tables.
Physical Fact Table	A table in the physical layer. It could also be an opaque view.

Table 40. Siebel Business Analytics Metadata Objects

Metadata	Description
Logical Dimension Table	Made up of one or more logical dimension table sources.
Logical Dimension Table Source	Made up of joins between multiple physical dimension tables.
Physical Dimension Table	A table in the physical layer. It could also be an opaque view. Might be shared with a physical fact table in the logical fact table source.
Complex Join	A join defined using complex expressions.
Key Join	A join defined using foreign key relationships.
Dimensional Hierarchy	A hierarchy defined on dimensions. Can have multiple keys per level.
Hierarchy Level	A level for each hierarchy. Should have a level key and related attributes.
Measure	A column in which data is derived from expressions involving other attributes.
Attribute	A column in the table.
Alias	A synonym for a table name. Will be used in the place of physical tables in the cube view. DDL will be generated to create these in the database.

Metadata Conversion Rules and Error Messages

This section explains the rules used to identify Siebel Business Analytics metadata that cannot be translated (converted) into XML format. These rules are necessary because IBM Cube Views does not support some of the metadata constructs that are allowed by Siebel Business Analytics.

Cube metadata in the XML file will be generated at the logical fact table source level. If a logical fact table source has an invalid logical dimension table source, then the logical dimension table source will be invalidated. If the logical fact table source is invalid, then all the logical dimension table sources that are linked to it will also be invalidated. Invalid analytics repository metadata elements will not be converted to cubes in the XML file.

When a rule is violated, the CubeViews Generator writes the error messages and the metadata that violated the rule to a log file. You specify the name of this log file in the parameter LOG_FILE_NAME in the NQSSConfig.INI file. For information about parameters in the Cube Views section of the NQSSConfig.INI file, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Table 41 on page 288 lists the rules used to validate analytics repository metadata elements, error messages written to the log file if the rule is violated, and an explanation of what caused the rule violation. The error messages help you determine why a particular Siebel Business Analytics metadata object was not exported to the XML file.

Table 41. Validation Rules for Metadata Elements

Rule	Message	Explanation
ComplexJoinFactsRule	[Fact Logical Table Source]Complex Physical Joins not supported %qn has a complex Join %qn between Physical Tables %qn and %qn	If the physical fact tables are connected through complex joins, the join is not supported. A complex join is defined as any join between two tables that do not have a foreign key relationship.
ComplexJoinDimensionsRule	[Dimension Logical Table Source]Complex Physical Joins not supported %qn has a complex Join %qn between Physical Tables %qn and %qn	If the dimension physical tables are connected through a complex join, then that join is not supported.
ComplexJoinFactDimRule	[Fact Logical Table Source -> Dimension Logical Table Source] Complex Physical Joins not supported. %qn has a complex Join %qn between Physical Tables %qn and %qn.	If a dimension physical table and a fact physical table are connected through a complex join, that join is not supported and the dimension table source is invalidated.
OpaqueViewFactRule	[Fact Logical table Source] Physical SQL Select Statements not supported. %qn uses the SQL Select Statement %qn.	When the physical fact table is generated by a SQL select statement, the logical fact table source that contains the table is invalidated. All logical dimension table sources connected to this logical fact table source are also invalidated. This construct allows subquery processing.
OpaqueViewDimensionsRule	[Dimension Logical table Source] Physical SQL Select Statements not supported. %qn uses the SQL Select Statement %qn.	When a physical dimension table is generated by a SQL select statement, the logical dimension table source containing that table is invalidated.
OuterJoinFactRule	[Fact Logical Table Source] Physical Outer Joins not supported. %qn has an outer join %qn between physical tables %qn and %qn.	If the logical fact table source has an outer join linkage, then that logical fact table source is invalidated and all logical dimension table sources linked to this source will also be invalidated.

Table 41. Validation Rules for Metadata Elements

Rule	Message	Explanation
OuterJoinDimRule	[Dimension Logical Table Source] Physical Outer Joins not supported. %qn has an outer join %qn between physical tables %qn and %qn.	If the logical dimension table source has an outer join linkage, that logical dimension table source is invalidated.
WhereClauseFactRule	[Fact Logical Table Source] WHERE clauses are not supported. %qn has a where condition %s.	If the fact table source uses a WHERE clause to filter the data that is loaded, then this table source is invalidated.
WhereClauseDimRule	[Dimension Logical Table Source] WHERE clauses are not supported. %qn has a where condition %s.	If the dimension table source uses a WHERE clause to filter the data that is loaded, this table source is invalidated.
TwoJoinFactDimRule	[Fact Logical Table Source -> Dimension Logical Table Source] Multiple Joins between sources not supported. %qn and %qn have at least the following joins : %qn, %qn.	If a physical fact table is linked to two dimension tables from the same dimension source (if the fact table is not exclusively linked to the most detailed table in the table source), the dimension table source is invalidated.
HiddenManyManyRule	[Fact Logical Table Source -> Dimension Logical Table Source] Join between (physical or logical?) fact and dimension is not on the most detailed table. %qn between %qn and %qn is not on the most detailed table %qn {Join name, facttable, dimtable}.	This is related to the TwoJoinFactDimRule. If the fact table is joined to a dimension table that is not the most detailed table in the table source, the dimension table source is invalidated.
ComplexMeasureRule	[Column] Complex Aggregation Rules not supported. %qn uses an aggregation rule of %s which is not supported.	The supported aggregations are SUM, COUNT, AVG, MIN, MAX, STDDEV, COUNT-DISTINCT, and COUNT.
CountDistinctMeasureRule	[Column] COUNT-DISTINCT Aggregation Rule not supported. %qn uses an aggregation rule of %s which is not supported.	COUNT-DISTINCT aggregation is not supported.
InvalidColumnLevelRule	[Level] Some columns that are part of the Primary Level Key are invalid. %qn has %qn as part of its primary key, when %qn has already been marked invalid.	COUNT-DISTINCT aggregation is not supported.

Table 41. Validation Rules for Metadata Elements

Rule	Message	Explanation
VariableBasedColumnRule	[Logical Table Source -> Column] Column uses a Variable in the Expression Column %qn uses a variable in its mapping.	COUNT-DISTINCT aggregation is not supported. The logical column uses repository and session variables in the expression.
OneFactToManyDimRule	[Fact Logical Table Source -> Dimension Logical Table Source] There must be a unique join path between the most detailed tables in the (logical or physical?) fact and the dimension. No join paths found between %qn and %qn (both physical table names). Found at least the following join paths: (%qn->%qn....), (%qn->%qn....)	Same as in TwoJoinFactDimRule or HiddenManyManyRule.
ManyMDTInFactRule	[Fact Logical Table Source] Fact Logical Table Source must have a unique most detailed table. %qn has at least the following most detailed tables : %qn,%qn.	A fact that has more than one table that is the most detailed table.
NoMeasureFactRule	[Fact Logical Table Source] Fact Logical Table Source does not have any Measures. %qn does not have any deployable measures.	A fact table does not have any measures because all the measures have been invalidated.
NoInActiveFactRule	[Fact Logical Table Source] Fact Logical Table Source is not marked Active.	A fact source is not active.
NoInActiveDimRule	[Dimension Logical Table Source] Dimension Logical Table Source is not marked Active.	A dimension source is not active.
NoAttributeInFactRule	[Fact Logical Table Source -> Column] Attribute found in Fact. %qn in a fact source %qn does not have an aggregation rule.	No attributes in the fact source.
NoMeasureInDimRule	[Dimension Logical Table Source -> Column] Measure found in Dimension. %qn in a dimension source %qn has an aggregation rule.	No measures in the dimension source.

Table 41. Validation Rules for Metadata Elements

Rule	Message	Explanation
VisibleColumns AttrRule	[Column] -> The run_as_user does not have visibility to this Logical Column. %qn is not accessible to the run_as_user %qn due to visibility rules.	A column does not have visibility for this user.
VisibleColumns MeasRule	[Column] -> The run_as_user does not have visibility to this Logical Column. %qn is not accessible to the run_as_user %qn due to visibility rules.	A column does not have visibility for this user.
MultiplePrimary KeysDimRule	[Dimension Logical Table Source] A Join uses an alternate key in the Dimension Logical Table Source. %qn between %qn and %qn in %qn uses the alternate key %qn.	A dimension physical table can contain only one primary key. It is joined to another dimension physical table using a different unique key and that join is invalid. IBM Cube Views does not accept any unique keys to be used for foreign joins and always requires the primary key.
MultiplePrimary KeysFactRule	[Dimension Logical Table Source] A Join uses an alternate key in the Dimension Logical Table Source. %qn between %qn and %qn in %qn uses the alternate key %qn.	A fact physical table can contain only one primary key. It is joined to another fact physical table using a different unique key and that join is invalid. IBM Cube Views does not accept any unique keys to be used for foreign joins and always requires the primary key.
MultiplePrimary KeysFactDimRule	[Fact Logical Table Source -> Dim Logical Table Source] A Join uses an alternate key between the Logical Table sources. %qn between %qn and %qn for sources %qn and %qn uses the alternate key %qn.	A fact physical table can contain only one primary key. It is joined to a dimension physical table using a different unique key and is invalid. IBM Cube Views does not accept any unique keys to be used for foreign joins and always requires the primary key.
NotDB2ExpressionAttrRule	[Dimension Logical Table Source -> Column] The Column contains an Expression not supported. %qn has expression %s which is not supported.	The attribute contains an expression not supported by IBM Cube Views. This includes metadata expressions that use DateTime functions (for example, CURRENT_DATE).

Table 41. Validation Rules for Metadata Elements

Rule	Message	Explanation
NotDB2ExpressionMeasRule	[Fact Logical Table Source -> Column] The Column contains an Expression not supported. %qn has expression %s which is not supported.	A measure contains an expression not supported by IBM Cube Views. This includes metadata expressions that use DateTime functions (for example, . CURRENT_DATE).
NoAttributeDimRule	[Dimension Logical Table Source] Dimension Logical Table Source does not have any attributes visible to the run_as_user. %qn can not be queried by user %qn since none of its attributes are visible.	A dimension does not have any attributes.

Process of Deploying Cube Metadata

The alias-SQL file generated by the CubeViews Generator should be executed before importing the XML file. The XML file generated by the CubeViews Generator contains the cube metadata in XML format. After importing the XML file into your DB2 database, you need to create materialized query tables.

NOTE: It is strongly recommended that you become familiar with IBM Cube Views and its tools before attempting to import the XML file. For more information, see your IBM documentation.

To deploy cube metadata, perform the following tasks in the order shown:

- 1 Executing the Alias-SQL File for IBM Cube Views on page 292
- 2 Importing the XML File on page 292
- 3 Guidelines for Creating Materialized Query Tables (MQTs) on page 294

Executing the Alias-SQL File for IBM Cube Views

This step is part of the [“Process of Deploying Cube Metadata” on page 292](#). You must execute the alias-SQL file before you import the XML file into your DB2 database. For more information, see your IBM documentation.

The alias-SQL file that is generated by the CubeViews Generator needs to be executed by a SQL client on the database where the data warehouse is located. When executed, it creates aliases (synonyms) for tables in the database.

Importing the XML File

This step is part of the [“Process of Deploying Cube Metadata” on page 292](#).

After you execute the alias-SQL file, you can import the XML file into the database. For more information, see your IBM documentation.

NOTE: It is strongly recommended that you become familiar with IBM Cube Views and its tools before attempting to import the XML file. For more information, see your IBM documentation.

You can import this file using the following IBM tools:

- **IBM OLAP Center (recommended).** For more information, see [“Guidelines for Importing the XML File Using the IBM OLAP Center” on page 293](#) and your IBM documentation.
- **IBM command-line client utility (db2mdapiclient.exe).** IBM ships this utility with DB2. For more information about using the command-line client utility, see your IBM documentation.
- **IBM DB2 Stored Procedure.** IBM Cube Views provides a SQL-based and XML-based application programming interface (API) that you can use to run single stored procedure to create, modify, and retrieve metadata objects. For more information, see your IBM documentation.

Guidelines for Importing the XML File Using the IBM OLAP Center

Using the IBM OLAP Center, you can import cube metadata into your DB2 database. The IBM OLAP Center provides wizards to help you import the file. For more information, see your IBM documentation.

To import the XML file, use the following guidelines:

- Using the IBM OLAP Center tool, connect to the DB2 database.
- In the Import Wizard, choose the XML file you want to import.
- If metadata exists that refers to database constructs that are not in the database, an error message appears.
- When the wizard asks for an import option, choose to replace existing objects.
- When you are returned to the IBM OLAP Center, you will see a diagram of the cube model.

Guidelines for Changing Cube Metadata After Importing the XML File

After you import the XML file, you might need to perform the following actions:

- Because Oracle's Siebel Data Warehouse does not store foreign keys as metadata, they will not exist in the converted metadata in the DB2 database. You need to use the IBM Referential Integrity Utility for IBM Cube Views to generate foreign key informational constraints. You can obtain this utility on the IBM Web site.
- You might encounter other issues such as foreign key join columns being nullable. You can use the following ways to solve this problem:
 - If data in these columns are not null, it is recommended that you convert these columns to not-null columns.
 - If data in these columns are null or you prefer not to convert the column data type even if the column data is not null, it is recommended that you modify the cube model using the following guidelines:

- In a fact-to-dimension join, you need to manually eliminate this dimension object from the converted cube model and create a degenerated dimension object consisting of the foreign key of this join.
- In a dimension-to-dimension join, you need to manually eliminate the dimension object that represents the primary-key side of the join from the converted cube model and create a degenerated dimension object consisting of the foreign key of this join.
- In a fact-to-fact join, you need to manually eliminate the fact object that represents the primary-key side of the join from the converted cube model and create a degenerated dimension object consisting of the foreign key of this join.
- No incremental metadata changes will be allowed by the Cube Generator. Schema changes require that you manually delete cube model metadata in the DB2 database and convert the Siebel Business Analytics metadata again. For example, if you need to make a change to a dimension in a cube in the Siebel Business Analytics metadata repository, you need to delete the cube model in the DB2 database, regenerate the XML file from the analytics repository, and import it into the DB2 database.
- You cannot delete metadata using the Siebel CubeViews Generator. The administrator needs to manually delete the cube model using the IBM OLAP Center.
- The IBM Statistics tool and IBM Optimization Advisor must be run periodically.

For more information, see your IBM documentation.

Guidelines for Creating Materialized Query Tables (MQTs)

This step is part of the [“Process of Deploying Cube Metadata” on page 292](#). For more information, see your IBM documentation.

After you import the cube metadata into the database, the administrator runs the IBM Optimization Advisor to generate SQL scripts and then execute those scripts to create the MQTs. The administrator needs to provide certain parameters to the IBM Optimization Advisor to get optimal results from the implementation. The IBM Optimization Advisor wizard analyzes your metadata and recommends how to build summary tables that store and index aggregated data for SQL queries. Running the IBM Optimization Advisor can help you keep the MQTs current. Additionally, you must refresh your database after each ETL.

To create MQTs, use the following guidelines:

- In the IBM OLAP Center, choose cube model that you want to optimize and then open the IBM Optimization Advisor wizard.
- Follow the instructions in the wizard, using the following table as a guide.

When asked for:	Choose:
Summary Tables	Choose Deferred (or Immediate) and provide a tablespace for the tables
Limitations	Choose an appropriate value for the optimization parameters. It is recommended to turn on the Data-sampling option.
SQL Scripts	Creation of the scripts needed to run to create the Summary tables. Choose the filename and locations

- When the IBM Optimization Advisor closes, the administrator needs to execute the SQL scripts to create the MQTs.

ODBC Conformance Level

The Analytics Server supports the following ODBC calls from client applications:

- SQLAllocConnect
- SQLAllocEnv
- SQLAllocStmt
- SQLBindCol
- SQLCancel
- SQLColumns
- SQLConnect
- SQLDescribeCol
- SQLDisconnect
- SQLDriverConnect
- SQLError
- SQLExecDirect
- SQLExecute
- SQLExtendedFetch
- SQLFetch
- SQLFreeConnect
- SQLFreeEnv
- SQLFreeStmt
- SQLGetConnectOption
- SQLGetCursorName
- SQLGetData
- SQLGetFunctions
- SQLGetInfo
- SQLGetStmtOption
- SQLGetTypeInfo
- SQLColAttributes
- SQLNumResultCols
- SQLPrepare

- SQLRowCount
- SQLSetConnectOption
- SQLSetStmtOption
- SQL Tables

Siebel Business Analytics ODBC supports full scrollable cursors with static, dynamic, forward only, and key set driven cursors.

Siebel Business Analytics ODBC supports asynchronous and synchronous processing and cancellation.

14 Using Variables in the Analytics Repository

You can use variables in a repository to streamline administrative tasks and modify metadata content dynamically to adjust to a changing data environment. The Administration Tool includes a Variable Manager for defining variables.

This section contains the following topics:

- [Using the Variable Manager on page 297](#)
- [Using Initialization Blocks on page 302](#)

Using the Variable Manager

The Variable Manager allows you to define variables. The Variable Manager dialog box has two panes. The left pane displays a tree that shows variables and initialization blocks, and the right pane displays details of the item you select in the left pane.

There are two classes of variables: repository variables and session variables.

- A repository variable has a single value at any point in time. There are two types of repository variables: static and dynamic. Repository variables are represented by a question mark icon.
- Session variables are created and assigned a value when each user logs on. There are two types of session variables: system and nonsystem.

System and nonsystem variables are represented by a question mark icon.

Initialization blocks are used to initialize dynamic repository variables, system session variables, and nonsystem session variables. The icon for an initialization block is a cube labeled *i*.

Using Repository Variables

A repository variable has a single value at any point in time. Repository variables can be used instead of literals or constants in expression builders in the Administration Tool. The Analytics Server will substitute the value of the repository variable for the variable itself in the metadata.

This section includes the following topics:

- [Static Repository Variables on page 297](#)
- [Dynamic Repository Variables on page 298](#)

Static Repository Variables

The value of a static repository value is initialized in the Variable dialog box. This value persists, and does not change until an administrator decides to change it.

Example

Suppose you want to create an expression to group times of day into different day segments. If Prime Time were one of those segments and corresponded to the hours between 5:00 PM and 10:00 PM, you could create a CASE statement like the following:

```
CASE WHEN "Hour" >= 17 AND "Hour" < 23 THEN 'Prime Time' WHEN... ELSE... END
```

where Hour is a logical column, perhaps mapped to a timestamp physical column using the date-and-time Hour(<<timeExpr>>) function.

Rather than entering the numbers 17 and 23 into this expression as constants, you could use the Variable tab of the Variable dialog box to set up a static repository variable named prime_begin and initialize it to a value of 17, and create another variable named prime_end and initialize it to a value of 23.

Using Variables in Expression Builders

After created, variables are available for use in expression builders. In an expression builder, click on the Repository Variables folder in the left pane to display all repository variables (both static and dynamic) in the middle pane by name.

To use a repository variable in an expression, select it and double-click. The expression builder will paste it into the expression at the active cursor insertion point.

Variables should be used as arguments of the function VALUEOF(). This will happen automatically when you double-click on the variables to paste them into the expression.

For example, the following CASE statement is identical to the one explained in the preceding example except that variables have been substituted for the constants.

```
CASE WHEN "Hour" >= VALUEOF("prime_begin") AND "Hour" < VALUEOF("prime_end") THEN  
'Prime Time' WHEN... ELSE... END
```

NOTE: You cannot use variables to represent columns or other repository objects.

Dynamic Repository Variables

You initialize dynamic repository variables in the same way as static variables, but the values are refreshed by data returned from queries. When defining a dynamic repository variable, you will create an initialization block or use a preexisting one that contains a SQL query. You will also set up a schedule that the Analytics Server will follow to execute the query and periodically refresh the value of the variable.

NOTE: When the value of a dynamic repository variable changes, all cache entries associated with a business model that reference the value of that variable will be purged automatically.

Each query can refresh several variables—one variable for each column in the query. You schedule these queries to be executed by the Analytics Server.

Example

Dynamic repository variables are very useful for defining the content of logical table sources. For example, suppose you have two sources for information about orders. One source contains recent orders and the other source contains historical data.

You need to describe the content of these sources on the Content tab of the Logical Table Source dialog box. Without using dynamic repository variables, you would describe the content of the source containing recent data with an expression such as:

```
Orders.OrderDates."Order Date" >= TIMESTAMP '2001-06-02 00:00:00'
```

This content statement will become invalid as new data is added to the recent source and older data is moved to the historical source. To accurately reflect the new content of the recent source, you would have to modify the fragmentation content description manually. Dynamic repository values can be set up to do it automatically.

Another suggested use for dynamic repository values is in WHERE clause filters of logical table sources, that are defined on the Content tab of the Logical Table Source dialog box.

The values of dynamic repository variables are set by queries defined in Variable Initialization blocks. When defining a dynamic repository variable, you create an initialization block or use a preexisting block that contains a query. You also set up a schedule that the Analytics Server will follow to execute the query and periodically refresh the value of the variable.

A common use of these variables in the Web is to set filters. For example, to filter a column on the value of the dynamic repository variable CurrentMonth set the filter to the Variable CurrentMonth.

About Session Variables

Session variables are like dynamic repository variables in that they obtain their values from initialization blocks. Unlike dynamic repository variables, however, the initialization of session variables is not scheduled. When a user begins a session, the Analytics Server creates new instances of session variables and initializes them.

Unlike a repository variable, there are as many instances of a session variable as there are active sessions on the Analytics Server. Each instance of a session variable could be initialized to a different value.

Session variables are primarily used when authenticating users against external sources such as database tables or LDAP servers. If a user is authenticated successfully, session variables can be used to set filters and permissions for that session. For a discussion of the use of session variables in setting up security, see [Chapter 16, "Security in Siebel Business Analytics."](#)

This section includes the following topics:

- [Using System Session Variables on page 300](#)
- [Using Nonsystem Session Variables on page 301](#)

For more information, see ["Creating New Variables" on page 301](#).

Using System Session Variables

System session variables are session variables that the Analytics Server and Siebel Business Analytics Web Server use for specific purposes. System session variables have reserved names, that cannot be used for other kinds of variables (such as static or dynamic repository variables and nonsystem session variables).

For information about using the GROUP system session variable in conjunction with the SA System subject area to provide group membership and external email addresses to Delivers, see [“Setting Up the Repository to Work with Delivers” on page 166](#).

NOTE: When you use these variables for Siebel Business Analytics Web, preface their names with NQ_SESSION. For example, to filter a column on the value of the variable LOGLEVEL set the filter to the Variable NQ_SESSION.LOGLEVEL.

Table 42 describes the available system session variables.

Table 42. System Session Variables

Variable	Description
USER	Holds the value the user enters as his or her logon name.
GROUP	<p>Contains the groups to which the user belongs. These are used by both the Analytics Server and Siebel Business Analytics Web Server.</p> <p>When a user belongs to multiple groups, separate the group names with semicolons. Do not delimit text (for example, do not surround the text with single or double quotes). Use a Varchar column in a database table to contain the group memberships.</p> <p>For example, if a user belonged to groups called Sales US, Sales UK, QA and Dev, and Doc, the text entered into a Varchar data type column in a database table would be:</p> <p style="text-align: center;">Sales US; Sales UK; QA and Dev; Doc</p> <p>Note: The Siebel Business Analytics Web administrator needs to make sure that the names of Web groups are different from any user IDs that will be used to log on to Siebel Business Analytics Web. If a user and a Web group share the same name, the user will receive an Invalid Account message when attempting to log on to Siebel Business Analytics Web.</p>
DISPLAYNAME	Used for Siebel Business Analytics Web. It contains the name that will be displayed to the user in the greeting in the Web interface. It is also saved as the author field for catalog objects. For internal analytics repository users (nondatabase users), this variable is populated with the user's full name.
PORTALPATH	Used for Siebel Business Analytics Web. It identifies the default dashboard the user sees when logging in (the user can override this preference after logged on).

Table 42. System Session Variables

Variable	Description
LOGLEVEL	<p>The value of LOGLEVEL (a number between 0 and 5) determines the Logging level that the Analytics Server will use for the user's queries.</p> <p>This system session variable overrides a variable defined in the Users object. If the Administrators Users object has a Logging level defined as 4 and the session variable LOGLEVEL defined in the repository has a default value of 0 (zero), the value of 0 applies.</p>
WEBGROUPS	Specifies additional groups specific to Siebel Business Analytics Web, if any. The use of Web groups provides a mechanism for more granular Web content control.
REQUESTKEY	Used for Siebel Business Analytics Web. Any users with the same nonblank request key will share the same Web cache entries. This tells Siebel Business Analytics Web that these users have identical content filters and security in the Analytics Server. Sharing Web cache entries is a way to minimize unnecessary communication with the server.
SKIN	Determines certain elements of the look and feel of the Siebel Business Analytics Web interface. The user can alter some elements of the user interface by picking a style when logged on to the Web. The SKIN variable points to a Siebel Business Analytics Web folder that contains the nonalterable elements (for example, graphics such as GIF files). Such directories begin with sk_. For example, if a folder were called sk_companyx, the SKIN variable would be set to companyx.
EMAIL	Contains the user's default email address for use with Answers. If the delivery feature of Answers is enabled, an email device using this address will be created for the user upon first log in. Users can override this address by changing their account settings in Siebel Business Analytics Web.

Using Nonsystem Session Variables

The procedure for defining nonsystem session variables is the same as for system session variables.

A common use for nonsystem session variables is setting user filters. For example, you could define a nonsystem variable called SalesRegion that would be initialized to the name of the user's sales region.

You could then set a security filter for all members of a group that would allow them to see only data pertinent to their region.

NOTE: When you use these variables for Siebel Business Analytics Web, preface their names with NQ_SESSION. For example, to filter a column on the value of the variable SalesRegion set the filter to the Variable NQ_SESSION.SalesRegion.

Creating New Variables

Use the following procedure to create a new variable.

To create a variable

- 1 From the Administration Tool menu bar, choose Manage > Variables.
- 2 In the Variable Manager dialog box, from the menu bar, choose Action > New, and then choose Repository Variable or Session Variable.
- 3 In the Variable dialog box, type a Variable name.

Names for all variables should be unique. The names of system session variables are reserved and cannot be used for other types of variables.

- 4 In the Variables dialog box, in the Type drop-down list, select the type of variable.

NOTE: The name of the dialog box depends on the type of variable that you select.

- For repository variables, choose Static Repository Variable or Dynamic Repository Variable.
- For session variables, choose Session Variable.

- 5 For session variables, you can select the following check box:

Enable any user to set the value

NOTE: This check box allows you to set the session variables after the initialization block has populated the value (at user login) by calling the ODBC store procedure NQSSetSessionValue(). For example, this allows non-administrators to set this variable for sampling.

- 6 For a dynamic repository variable or a session variable, use the Initialization Block drop-down list to select an initialization block that will be used to refresh the value on a continuing basis.

For more information, see [“Creating and Editing Initialization Blocks” on page 306](#).

- 7 To add a Default initializer value, perform one of the following steps:

- To use the Expression Builder, click the ellipsis button to the right of the Default initializer work space. For more information about creating the value, see [“SQL Logical Operators” on page 374](#).
- Type the value into the Default initializer text box.

For static repository variables, the value you specify in the Default initializer window persists, and will not change unless you decide to change it. If you initialize a variable to a character string, enclose the string in single quotes (').

- 8 Click OK.

Using Initialization Blocks

Initialization blocks are used to initialize dynamic repository variables, system session variables, and nonsystem session variables. For example, the NQ_SYSTEM initialization block is used to refresh system session variables.

An initialization block contains the SQL that will be executed to initialize or refresh the variables associated with that block. The SQL must reference physical tables that can be accessed using the connection pool specified in the Connection Pool field in the Initialization Block dialog box.

If you want the query for an initialization block to have database-specific SQL, you can select a database type for that query. If a SQL initialization string for that database type has been defined when the initialization block is instantiated, this string will be used. Otherwise, a default initialization SQL string will be used.

CAUTION: By default, when you open the Initialization Block dialog box for editing in online mode, the initialization block object is automatically checked out. While the initialization block is checked out, the Analytics Server may continue to refresh the value of dynamic variables refreshed by this initialization block, depending on the refresh intervals that are set. When you check the initialization block in, the value of the dynamic variables is reset to the values shown in the Default initializer. If you do not want this to occur, use the Undo Check Out option.

Table 43 on page 303 lists some of the initialization blocks common to all Siebel Analytics applications and their purposes. Initialization blocks that are specific to each Siebel Analytics applications area are not listed here. For example, the LAST_SYND_DATES block sets certain dates for Pharmaceutical Siebel Business Analytics and can be ignored by other applications that do not use syndicated data.

Table 43. Initialization Blocks and Their Purposes

Initialization Block	Purpose
Authorization	Calculates user's responsibilities from the database.
Authentication	Authenticates against the database and verifies the user exists as a operational application user.
External Metadata Strings	Gets the translated value of metadata strings for the user's locale. This initialization block is critical to Intelligence Dashboards in international deployment.
LOCALE	Sets the user's locale specification in the Siebel Analytics Server.
Login Properties	Gets user's login properties, such as full name, from the database. This initialization block also sets the logging level of the users. By default, the Analytics log level has been set to 0 for all users. If you wish to generate Analytics query logs for all users, this initialization block should be updated by changing the default value as well as the value in the initialization SQL.
Default System Language ID	Sets the variable OLTP_LANG_ID by querying the operational application database.
Orgs for Org-based Security	Queries the operational application database to get the organization membership for each user. It sets the variable ORGANIZATION.
Primary Owner ID	Sets the primary owner ID based on the user login ID.
Primary Position ID	Queries the operational application database to set the variable PRIMARY_POSTN_ID.
Warehouse Refresh Date	Sets several time-based variables such as CURRENT_YEAR.

Table 43. Initialization Blocks and Their Purposes

Initialization Block	Purpose
ETL Run Date	Retrieves the ETL run date.
ETL Default Currency	Retrieves the default currency.

Initializing Dynamic Repository Variables

The values of dynamic repository variables are set by queries defined in the Initialization string field of the Initialization Block dialog box. You also set up a schedule that the Analytics Server will follow to execute the query and periodically refresh the value of the variable. If you stop and restart the Analytics Server, the server automatically executes the SQL in repository variable initialization blocks, reinitializing the repository variables.

The Analytics Server logs all SQL queries issued to retrieve repository variable information in the NQQuery.log file when the administrator logging level is set to 2 or higher. You should set the logging level to 2 for the administrator user ID to provide the most useful level of information. The default location for the NQQuery.log file is the Log folder in the Analytics Server software installation folder (\Siebel Analytics). For more information about user-level logging, see [“Administering the Query Log” on page 242](#).

Initializing Session Variables

As with dynamic repository variables, session variables obtain their values from initialization blocks. Unlike dynamic repository variables, session variables are not updated at scheduled time intervals. Instead, the Analytics Server creates new instances of those variables whenever a user begins a new session. The values remain unchanged for the session's duration.

The Analytics Server logs all SQL queries issued to retrieve session variable information if Logging level is set to 2 or higher in the Security Manager User object or the LOGLEVEL system session variable is set to 2 or higher in the Variable Manager.

The default location for the NQQuery.log file is the Log folder in the Analytics Server software installation folder (\Siebel Analytics). For more information about user-level logging, see [“Administering the Query Log” on page 242](#).

Row-Wise Initialization

The row-wise initialization feature allows you to create session variables dynamically and set their values when a session begins. The names and values of the session variables reside in an external database that you access through a connection pool. The variables receive their values from the initialization string that you type in the Initialization Block dialog box.

For example, you want to create session variables using values contained in a table named RW_SESSION_VARS. The table contains three columns: USERID, containing values that represent users' unique identifiers; NAME, containing values that represent session variable names; and VALUE, containing values that represent session variable values.

The content of the table is as follows:

USERID	NAME	VALUE
JOHN	LEVEL	4
JOHN	STATUS	FULL-TIME
JANE	LEVEL	8
JANE	STATUS	FULL-TIME
JANE	GRADE	AAA

You create an initialization block and select the Row-wise initialization check box (see [“Creating and Editing Initialization Blocks” on page 306](#)).

For the initialization string, you type the following SQL statement:

```
select NAME, VALUE
from RW_SESSION_VARS
where USERID='VALUEOF(NQ_SESSION.USERID)'
```

NQ_SESSION.USERID has already been initialized using another initialization block.

The following session variables are created:

- When John connects to the Analytics Server, his session will contain two session variables from row-wise initialization: LEVEL, containing the value 4; and STATUS, containing the value FULL-TIME.
- When Jane connects to the Analytics Server, her session will contain three session variables from row-wise initialization: LEVEL, containing the value 8; STATUS, containing the value FULL-TIME; and GRADE, containing the value AAA.

Initializing a Variable with a List of Values

You can also use the row-wise initialization feature to initialize a variable with a list of values. You can then use the SQL IN operator to test for values in a specified list.

Example: Using the table values in the previous example, you would type the following SQL statement for the initialization string:

```
select 'LIST_OF_USERS', USERID
from RW_SESSION_VARS
where NAME='STATUS' and VALUE='FULL-TIME'
```

This SQL statement populates the variable LIST_OF_USERS with a list, separated by colons, of the values JOHN and JANE; for example, JOHN:JANE. You can then use this variable in a filter, as shown in the following WHERE clause:

```
where TABLE.USER_NAME = valueof(NQ_SESSION.LIST_OF_USERS)
```

The variable LIST_OF_USERS contains a list of values, that is, one or more values. This logical WHERE clause expands into a physical IN clause, as shown in the following statement:

where TABLE.USER_NAME in (' JOHN' , ' JANE')

Creating and Editing Initialization Blocks

When you create SQL and submit it directly to the database, bypassing Analytics Server (for example when creating initialization blocks), you should test the SQL using the Test button. If the SQL contains an error, the database returns an error message.

It is recommended to create a dedicated connection pool for initialization blocks. For more information, see [“Creating or Changing Connection Pools” on page 68](#).

Use the instructions in this section to create a new initialization block or to edit properties of an existing initialization block. [Table 44 on page 307](#) contains descriptions of some of the elements in the Initialization Block dialog box. For more information about initialization blocks, see [“Using Initialization Blocks” on page 302](#).

To create or edit an initialization block

- 1 From the Administration Tool menu bar, select Manage > Variables.
- 2 In the Variable Manager dialog box, from the Action menu, choose New > Initialization Block.
- 3 In the Initialization Block dialog box, complete the fields using [Table 44 on page 307](#) as a guide.
- 4 In the General tab, type a name for the block. (The NQ_SYSTEM initialization block name is reserved.)
- 5 From the drop-down list, select one of the following values:
 - associate with repository variable
 - associate with session variable

NOTE: If you select associate with session variable, you can use the row-wise initialization feature to dynamically create session variables and set their values when a session begins. For more information, see [“Row-Wise Initialization” on page 304](#).
- 6 To use row-wise initialization, select the Row-wise initialization check box.
The Cache Variables check box is automatically selected.
- 7 Specify a refresh interval in the Refresh time area (repository variables and database data sources only).
 - a From the Start on drop-down list, select a date and time.
 - b In the Refresh interval field, type the periodic duration of the refresh interval.
 - c From the Refresh interval drop-down list, select days, hours, or minutes.
- 8 In the Data Source Connection drop-down list, select Database or select LDAP if you are authenticating LDAP users.
- 9 If you selected Database in the Data Source Connection drop-down list, perform the following steps:

a Select the connection pool associated with the database where the target information is located by clicking Browse.

b In the Browse dialog box, select the connection pool and click OK.

NOTE: Select a connection pool before typing an initialization string.

c For initialization strings that are database-specific, in the Initialization string drop-down list, select the type of database.

d In the Initialization string text box, type the SQL initialization string needed to populate the variables.

CAUTION: If you have not selected a connection pool before typing the initialization string, you will receive a message prompting you to select the connection pool.

10 If you selected LDAP in the Data Source Connection area, perform the following steps:

a Click Browse to select an existing LDAP Server or click New to open the General tab of the LDAP Server dialog box and create an LDAP Server.

b Click OK to return to the Initialization Block dialog box.

The LDAP server name and the associated domain identifier appear in the Name and Domain identifier columns.

11 (Optional) Click Test in the Data Source Connection area.

12 In the Set value for the variables dialog box, verify the information is correct, and then click OK.

13 In the View Data from Table dialog box, type the number of rows and the starting row for your Query, and then click Query.

The Results dialog box lists the variables and their values.

NOTE: You should test the SQL using the Test button or an SQL tool such as Oracle's Siebel Analytics Client utility. If you use an SQL tool, be sure to use the same DSN or one set up identically to the DSN in the specified connection pool.

Table 44. Initialization Block Dialog Box Description

Field	Description
association drop-down list	An initialization block can be associated with repository or session variables for database data source connections but can be associated with only session variables for LDAP data source connections.
Enable (check box)	When selected, enables the initialization block. In the Variables Manager, when you right-click an existing initialization block, the menu contains a Disable or Enable toggle value, allowing you to change this property without having to open the Initialization Block dialog box.
Row-wise initialization (check box)	Can be selected for session variables only. For more information, see "Row-Wise Initialization" on page 304 .

Table 44. Initialization Block Dialog Box Description

Field	Description
Refresh time Start on Refresh interval	<p>The Refresh time area only applies to repository variables. It is unavailable for session variables.</p> <p>You can specify the day, the month, the day of the month, the year, and the hours, minutes and seconds.</p>
Data Source Connection	<p>Choose Database or LDAP.</p> <p>If you select Database as the data source connection, the values returned by the database in the columns in your SQL will be assigned to variables. The order of the variables and the order of the columns will determine which columns are assigned to which variables.</p>
Test (button)	In Online editing mode, Initialization Block tests will not work with connection pools set to use :USER and :PASSWORD as the user name and password. In offline mode, the Set values for variables dialog box appears so that you can populate :USER and :PASSWORD.
Initialization string: (drop-down list)	If you use an initialization string that is database-specific, you would select the type of database. Otherwise, use Default.
Initialization string: (SQL text box)	<p>At run time, if an initialization string for the database type has been defined, this string will be used. Otherwise, the default initialization SQL for the database type will be used.</p> <p>If you select Database in the Data Source Connection area, the SQL used to refresh the variable must reference physical tables that can be accessed through the connection pool specified in the Connection Pool field. The tables do not have to be included in the physical layer of the metadata.</p> <p>The order of the variables and the order of the columns determine which columns are assigned to each variable.</p> <p>For examples of SQL initialization strings, see “Examples of Initialization String SQL Statements” on page 309.</p>
Cache variables (check box)	<p>The Cache variables check box is automatically selected when you select the Row-wise initialization check box. Selecting the cash variables option directs the Analytics Server to store the results of the query in a main memory cache.</p> <p>The Analytics Server uses the cached results for subsequent sessions. This can reduce, often significantly, session startup time. However, the cached results may not contain the most current session variable values. If every new session needs the most current set of session variables and their corresponding values, you clear this check box.</p>
Connection pool	Used for Database data source connections.

Examples of Initialization String SQL Statements

The examples in this section are of initialization strings that might be used with Delivers.

Example of an SQL Statement When Site Uses Delivers

```
select username, groupname, dbname, schemaname from users
where username=: USER'
NQS_PASSWORD_CLAUSE(and pwd=: PASSWORD')NQS_PASSWORD_CLAUSE
```

This SQL contains two constraints in the WHERE clause:

':USER' (note the colon and the single quotes) equals the ID the user types when logging in.

':PASSWORD' (again, note the colon and the single quotes) is the password the user enters. This is another system variable whose presence is always assumed when the USER system session variable is used. You do not need to set up the PASSWORD variable, and you can use this variable in a database connection pool to allow passthrough login using the user's user ID and password. You can also use this variable in a SQL statement if you so desire.

When using external table authentication with Delivers, the portion of the SQL statement that makes up the :PASSWORD constraint needs to be embedded between NQS_PASSWORD_CLAUSE clauses.

The query will return data only if the user ID and password match values found in the specified table. You should test the SQL statement outside of the Analytics Server substituting valid values for the USER and PASSWORD variables and removing the NQS_PASSWORD_CLAUSE clause.

For more information, see ["About Delivers and Database Authentication" on page 340](#).

Example of an SQL Statement When Site Does Not Use Delivers

```
select username, groupname, dbname, schemaname from users
where username=: USER'
and pwd=: PASSWORD'
```

This SQL statement contains two constraints in the WHERE clause:

':USER' (note the colon and the single quotes) is the ID the user enters when the user logged in.

':PASSWORD' (again, note the colon and the single quotes) is the password the user enters. This is another system variable whose presence is always assumed when the USER system session variable is used. You do not need to set up the PASSWORD variable, and you can use this variable in a database connection pool to allow passthrough login using the user's user ID and password. You can also use this variable in a SQL if you so desire.

The query will return data only if the user ID and password match values found in the specified table. You should test the SQL statement outside of the Analytics Server, substituting valid values for the USER and PASSWORD variables.

Tasks Using the Initialization Block Dialog Box—Variable Tab

The Variables tab can list several variables in the Variables column, and the initialization block SQL in the Default initializer list can list multiple columns in the SELECT statement. The variables will be initialized in the same order that the columns are listed in the SQL statement; for example, the value returned in the n th column will be assigned to the n th variable in the list.

For repository variables, when you open a repository in online mode, the value shown in the Default initializer column in the Variable tab of the Initialization Block dialog box is the current value of that variable as known to the Analytics Server.

NOTE: The number of associated variables can be different from the number of columns being retrieved. If there are fewer variables than columns, extra column values are ignored. If there are more variables than columns, the additional variables are not refreshed (they retain their original values, whatever they may be). Any legal SQL can be executed using an initialization block, including SQL that writes to the database or alters database structures, assuming the database permits the user ID associated with the connection pool to perform these actions.

For more information about initialization blocks, see [“Using Initialization Blocks” on page 302](#).

To access the Initialization Block dialog box

- 1 In the Administration Tool, select Manage > Variables.
- 2 In the Variable Manager dialog box, select Action > New > Initialization Block.

To reorder a variable

- 1 In the Variables list, select the variable you want to reorder.
- 2 Click Up and Down to reorder the list.

To add a new variable to refresh with this block

- 1 Click the New button to open the Variable tab of the Variable dialog box.
- 2 Complete the Variable tab, and then click OK.

To edit a variable refreshed by this block

- 1 Select the variable to edit and click Edit to open the Variable tab of the Variable dialog box.
- 2 Make your changes, and then click OK.

To link a block with a variable not already refreshed by an initialization block

- 1 Click the Link button.
The Browse dialog box appears.

- 2 Select the variable to be refreshed by this initialization block.
- 3 Click OK to close the Browse dialog box and return to the Initialization Block dialog box.

To remove a variable's association with this block

- Select the variable, and then click the Remove button.

Default Initializer

For repository variables, when you open a repository in online mode, the value shown in the Default initializer column of the Initialization Block—Variables tab is the *current* value of that variable as known to the Analytics Server.

Initialization When the Analytics Server Starts

If you stop and restart the Analytics Server, the server will automatically execute the SQL in repository variable initialization blocks, reinitializing the repository variables.

Execution Precedence

When a repository has more than one initialization block, you can set the order of precedence of the blocks. For example, suppose a repository has two initialization blocks, A and B. When you define block B, you can specify that block A will execute before block B. In effect, this causes block A to execute according to block B's schedule, in addition to its own.

The order of precedence is shown on the Execution Precedence tab of the Initialization Block dialog box.

The Execution Precedence tab displays the initialization blocks that will be executed before the block you have open.

To set the execution order

- Click Add and select the block from the Browse dialog box.

To remove a block

- Select the block you want to remove, and then click Remove.

15 Clustering Analytics Servers

This section describes the Cluster Server feature and provides instructions for setting up and configuring the clustering of multiple servers.

About the Cluster Server Feature

The Cluster Server feature allows up to 16 Analytics Servers in a network domain to act as a single server. Servers in the cluster share requests from multiple Siebel Business Analytics clients, including Answers and Delivers.

The Cluster Controller is the primary component of the Cluster Server feature. It monitors the status of resources in a cluster and performs session assignment as resources change. It also supports detection of server failures and failover for ODBC clients of failed servers.

Components of the Cluster Server Feature

This section describes the components of the Cluster Server feature.

Server Manager (UNIX Only)

The Server Manager is a small module that is part of the `nqscomgateway.exe` process. It is only used while running on UNIX. It controls the start/stop/restart of the `nqsserver.exe` process. In the `NQClusterConfig.INI` file the `SERVER_MANAGER_PORT` parameter specifies the port number to use for communication.

Primary Cluster Controller

The role of the primary Cluster Controller is to monitor the operation of the servers in the cluster and to assign sessions within the cluster. The primary Cluster Controller can reside on the same machine as a Analytics Server in the cluster or on another machine that is on the same subnet as the cluster. A machine can host one Analytics Server, one Cluster Controller, or one of each.

In the `NQClusterConfig.INI` file, the parameter `PRIMARY_CONTROLLER` specifies the machine that hosts the primary Cluster Controller.

Secondary Cluster Controller

The secondary Cluster Controller assumes the role of the primary Cluster Controller if the primary is unavailable. The secondary Cluster Controller can reside on the same machine as a Analytics Server in the cluster or on another machine that is on the same subnet as the cluster.

In the NQClusterConfig.INI file, the parameter SECONDARY_CONTROLLER specifies the machine that will host the secondary Cluster Controller. It must be different from the machine that hosts the primary Cluster Controller. Specifying a secondary Cluster Controller is optional. However, if the primary Cluster Controller is unavailable and the secondary Cluster Controller has not been configured, the cluster will not operate.

Master Server

The master server is a clustered Analytics Server to which the Administration Tool connects for online repository changes. In the NQClusterConfig.INI file, the parameter MASTER_SERVER specifies the Analytics Server that functions as the master server.

Repository Publishing Directory

This directory is shared by all Analytics Servers participating in a cluster. It holds the master copies of repositories edited in online mode. The clustered Analytics Servers examine this directory upon startup for any repository changes. The directory typically resides on a shared file system visible to all servers in the cluster. The master server must have read and write access to this publishing directory.

In the NQSConfig.INI file, the REPOSITORY_PUBLISHING_DIRECTORY parameter specifies the location of the repository publishing directory.

Cluster Manager

The Cluster Manager is available in the Administration Tool when a repository is open in online mode. It allows the administrator to monitor and manage the operations and activities of the cluster.

Implementing the Cluster Server Feature

These are the high-level steps to implement the Cluster Server feature:

- 1 Install Siebel Business Analytics, including the Cluster Controller component.
- 2 Set parameters in the NQSConfig.INI file.
- 3 Set parameters in the NQClusterConfig.INI file.
- 4 Set up the Siebel Business Analytics ODBC data source for clustering.
- 5 Copy the NQClusterConfig.INI file to the Cluster Controllers and Analytics Server in the cluster.
- 6 Start the machines in the cluster.

For detailed information about installing and configuring this feature, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Installing the Cluster Server Feature

The Cluster Controller is the only component of the Cluster Server feature that needs to be installed. To install this component, you select the Custom option in the Installation Wizard for machines that will host Cluster Controllers and then select Siebel Analytics Cluster.

You should review the hardware and software requirements for the Cluster Controller before you install it; see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Setting Parameters in the NQConfig.INI File

In the Server section of the NQConfig.INI file, there are three parameters you need to set for a Analytics Server that will participate in a cluster. A brief description of each parameter follows:

- CLUSTER_PARTICIPANT. Specifies whether the Analytics Server is a member of a cluster.
- REPOSITORY_PUBLISHING_DIRECTORY. Specifies the location of the repository publishing directory shared by all Analytics Servers participating in the cluster.
- REQUIRE_PUBLISHING_DIRECTORY. Specifies whether the repository publishing directory needs to be available in order for the Analytics Server to start up and join the cluster.

The NQConfig.INI file is located in the Config directory in the Siebel Business Analytics software installation folder. For more information about these parameters, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Setting Parameters in the NQClusterConfig.INI File

The NQClusterConfig.INI file contains the cluster configuration parameters. The Analytics Server reads this file after it reads the NQConfig.INI file (when CLUSTER_PARTICIPANT is set to YES in the NQConfig.INI file). Cluster Controllers also read this file. A brief description of each parameter follows:

- ENABLE_CONTROLLER—Specifies whether the Cluster Controller functionality is enabled. The NO setting allows the administrator to temporarily disable a Cluster Controller if, for example, the machine is being serviced. This parameter only applies to the Cluster Controller on the machine on which it resides.
- PRIMARY_CONTROLLER—Specifies the machine that acts as the primary Cluster Controller.
- SECONDARY_CONTROLLER—Specifies the machine that will take over the responsibilities of the primary Cluster Controller if the primary becomes unavailable. Specifying a secondary Cluster Controller is optional.
- SERVERS—Specifies the Analytics Servers that belong to the cluster. A cluster can contain a maximum of 16 Analytics Servers. A server can belong to only one cluster.
- MASTER_SERVER—Specifies the Analytics Server to which the Administration Tool connects for online repository changes. Online repository changes are published to the location specified by the REPOSITORY_PUBLISHING_DIRECTORY parameter in the NQConfig.INI file.
- SERVER_POLL_SECONDS—Specifies the frequency of heartbeat messages between the Cluster Controller and each server in the cluster.

- **CONTROLLER_POLL_SECONDS**—Specifies the frequency of heartbeat messages between the primary Cluster Controller and the secondary Cluster Controller if one is defined.

The NQClusterConfig.INI file is located in the Config directory in the Siebel Business Analytics installation folder. For more information about these parameters, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Configuring the Analytics ODBC Data Source Name

All clients, including Siebel Business Analytics Web clients, need to have a clustered data source name (DSN) configured in order to communicate with a cluster. You set up a DSN by using the Siebel Analytics DSN Configuration wizard, described in [“Connectivity and Third-Party Tools” on page 279](#).

Copying the NQClusterConfig.INI File

A configured NQClusterConfig.INI file needs to reside in the Config directory of every Analytics Server and Cluster Controller that is to participate in the cluster.

For detailed instructions on configuring the NQClusterConfig.INI file, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Starting Cluster Controllers and Analytics Servers

When you are using the Administration Tool and have a repository open in online mode, you can use the Cluster Manager to monitor and manage the operations of the cluster, including starting and stopping Analytics Servers and Cluster Controllers. However, opening a repository in online mode does not automatically start a clustered Analytics Server or a Cluster Controller; therefore, you must start one Analytics Server and one Cluster Controller manually. You can then use the Cluster Manager to start additional clustered servers.

NOTE: On UNIX each server needs to be started manually. After the servers are running, you can use the Cluster Manager to start and stop the Analytics Server.

To manually start a Analytics Server and Cluster Controller in Windows

- 1 Navigate to the Services window by selecting Start > Programs > Administrative Tools > Services.
- 2 Right-click Siebel Analytics Server and click Start.
- 3 Right-click Siebel Analytics Cluster and click Start.

To start the Analytics Server and Cluster Controller from the Command window

- Open a Command window and type the following:

```
net start "Siebel Analytics SERVER"
```

```
net start "Siebel Analytics CLUSTER"
```

NOTE: You can also use a third-party tool designed for remote service manipulation.

After you start the Analytics Servers and Cluster Controller, use a text editor to examine the log files `NQServer.log` and `NQCluster.log` in the Log directories, and verify that all machines started without errors and joined the operational cluster configuration successfully. If the log files indicate errors, correct the errors and restart the servers.

To start the Analytics Server and Cluster Controller from the command line on UNIX

- 1 Navigate to a command window (xterm).
- 2 In the Command window, type the following commands (the commands will be slightly different if using `csh`):

```
cd INSTALLDIR/setup
./run-sa.sh start
./run-ccs.sh start
```

Chronology of a Cluster Operation

This section provides an overview of the Analytics Cluster Server startup process.

- 1 As each Analytics Server starts, it reads its `NQSConfig.INI` file. If a server detects a syntax error while reading the file, it logs the error to its `NQServer.log` file in its Log directory. All syntax errors have to be corrected for startup to continue.
- 2 Each Analytics Server reads its `NQClusterConfig.INI` file when `CLUSTER_PARTICIPANT` is set to YES in the `NQSConfig.INI` file. Cluster Controllers also read this file. If a Analytics Server detects a syntax error while reading the file, it logs the error to its `NQServer.log` file. If a Cluster Controller detects an error while reading the file, it logs the error to its `NQCluster.log`. If a machine is hosting both a Analytics Server and a Cluster Controller, messages will be written to both logs. All syntax errors have to be corrected for startup to continue.
- 3 The Analytics Server verifies the presence of an active, operational Cluster Controller. If the machine hosting the primary Cluster Controller is not available, the Analytics Server will contact the machine designated as the secondary if a secondary has been defined. If no Cluster Controller can be located, cluster startup will fail.
- 4 The primary and secondary Cluster Controllers begin to exchange heartbeat messages. (This step is omitted when no secondary Cluster Controller is defined.)
- 5 The Analytics Server verifies whether the repository publishing directory is available. If the repository publishing directory is not available, the action each server takes depends on the setting for the `REQUIRE_PUBLISHING_DIRECTORY` parameter in its `NQSConfig.INI` file. When set to YES, if the publishing directory is not available at startup or if an error is encountered while the server is reading any of the files in the directory, an error message is logged in the `NQServer.log` file and the server shuts down. When set to NO, the server joins the cluster and a warning message is logged in the `NQServer.log` file, but any online repository updates are not reflected in the server's Repository directory.

- 6 The primary and secondary Cluster Controllers begins to exchange heartbeat messages with each server that is to participate in the cluster. The connection status is logged in the NQServer.log files of all servers in the SERVERS list. Messages are also logged in the Cluster Controller's NQCluster.log file. Any servers with connection problems are not be allowed to join the cluster. If the server defined as the MASTER_SERVER for online repository is not available, no repository editing in online mode will be possible.
- 7 As each Analytics Server in the cluster is started, it examines the repository publishing directory for any updated repositories. This is done by comparing the date and timestamps. (The administrator is responsible for making sure that the time of day clocks are synchronized across all Analytics Servers and Cluster Controllers.) If a server detects a newer version of an existing repository, it copies the repository to its own Repository directory. A server will not detect the presence of any new repositories; a new repository must be manually propagated to all clustered servers when it is initially created. After that, online changes are detected at subsequent startups of each server.
- 8 When the Cluster Controller assigns a session to a particular Analytics Server, the server communicates with the back-end database using the connection defined in the Connection Pool dialog box for the database. Clustered servers do not share a common connection pool.
- 9 If an Analytics Server determines it can satisfy all or a portion of a query from its cache file, it will do so. Clustered servers do not share a common cache.

Using the Cluster Manager

The Cluster Manager allows you to monitor, analyze, and manage the operations of a cluster. It provides status, cache, and session information about the servers and controllers that make up a cluster. It is available only when the Administration Tool is connected to a clustered DSN.

NOTE: If all Cluster Controllers or Analytics Servers in the cluster are currently stopped or offline, you cannot access the Cluster Manager to start them. You must manually start one Cluster Controller (generally, the primary) and one Analytics Server; the others can then be started from the Cluster Manager window.

Cluster Manager GUI

The Cluster Manager Graphical User Interface (GUI) has two panes: the Explorer pane on the left side and the Information pane on the right side. The Explorer pane displays hierarchical information about the controllers and servers that make up a cluster. The Information pane shows detailed information about an item selected in the Explorer pane.

To access the Cluster Manager

- 1 In the Administration Tool, open a repository in online mode.
- 2 Select Manage > Clusters.

To refresh the display

- The Cluster Manager window refreshes every minute by default. You can change this value by selecting Refresh > Every and selecting another value from the list.
- To refresh the display at any time, make sure the Cluster Manager is the active window and press F5, or select Refresh > Now. This retrieves the most current information for the cluster.

Viewing Cluster Information

The section describes how to view status, cache, and session information about a cluster and the meaning of the information provided.

Status Information

The Status view is automatically displayed when you first open the Cluster Manager window. You can also access the Status view by selecting View > Status in the Cluster Manager window.

The categories of information displayed in the Information pane may vary depending on the server to which Administration Tool is connected. [Table 45](#) describes categories that may appear.

Table 45. Status View Columns

Column	Description
Name	The name of the machine hosting the Analytics Server or Cluster Controller.
Type	When Clusters is selected in the Explorer pane, this field is available. There are two types: <ul style="list-style-type: none"> ■ Controller - The object is a Cluster Controller. ■ Server - The object is a Analytics Server.
Role	The role of the object in the cluster: <ul style="list-style-type: none"> ■ Controlling - A Cluster Controller that is currently assigned the responsibility for control of the cluster. ■ Primary - The primary Cluster Controller. This role is not displayed if the primary Cluster Controller is currently the controlling Cluster Controller. ■ Secondary - The secondary Cluster Controller. This role is not displayed if the secondary Cluster Controller is currently the controlling Cluster Controller. ■ Clustered server - A Analytics Server that is a member of the cluster. This role is not displayed for the clustered server defined as the master server. ■ Master - The clustered server that the Administration Tool connects to for editing repositories in online mode.

Table 45. Status View Columns

Column	Description
Status	<p>The status of the object in the cluster:</p> <ul style="list-style-type: none"> ■ Online - The Cluster Controller or Analytics Server is online. For cluster controllers, this means the controller can accept session requests and assign them to available servers within the cluster. For clustered servers, this means that the server may be assigned sessions by the cluster controller. ■ Quiesce - This status is applicable to clustered servers only. This means that any activity in progress on outstanding sessions will be allowed to complete before the server transitions to Offline status. ■ Offline - The Cluster Controller or Analytics Server is offline. For Cluster Controllers, this means the controller cannot accept session requests or assign sessions to available servers within the cluster. For clustered servers, this means that the server is not communicating with the controlling Cluster Controller and cannot accept sessions assigned by the controlling Cluster Controller. If the server subsequently becomes available, it will be allowed to participate in the cluster. If you want to stop the Cluster Controller or clustered server after quiescing it, you need to issue the Stop command. ■ Forced Offline - This status applies to clustered servers only. The Analytics Server has been stopped. This is identical to the offline status, except that if the Analytics Server comes back online, it will not be assigned requests. The server will remain in this state until the Start command is issued against this server from the Administration Tool Cluster Manager or both Cluster Controllers are shut down and restarted.
Start Time	The timestamp showing when the Cluster Controller or Analytics Server was last started. This field will be blank if the Cluster Controller or clustered server is offline.
Last Reported Time	The time the Cluster Controller or Analytics Server communicated with the Controlling Cluster Controller. If the server or controller is offline, this field may be blank.
Sessions	This field is available when either Servers or an individual server is selected in the Explorer pane. It shows the number of sessions currently logged on to a clustered server.

Cache Information

The Cache view is available in the Cluster Manager window if caching is enabled.

The categories of information and their display sequence are controlled by your Options settings. [Table 46](#) describes categories that may appear.

Table 46. Cache View Columns

Column	Description
User	ID of the user who submitted the query that resulted in the cache entry.
Created	Time the cache entry's result set was created.
Last used	Last time the cache entry's result set satisfied a query. (After an unexpected shutdown of a Analytics Server, the Last used time may temporarily have a stale value, that is, older than the true value.)
Creation elapsed time	Time, in milliseconds, needed to create the result set for this cache entry.
Row count	Number of rows generated by the query.
Row size	Size of each row (in bytes) in this cache entry's result set.
Full size	Full size is the maximum size used, considering variable length columns, compression algorithm, and other factors. The actual size of the result set will be smaller than Full size. Time, in seconds, needed to create the result set for this cache entry.
Column count	Number of columns in each row of this cache entry's result set.
Use count	Number of times this cache entry's result set has satisfied a query (since Analytics Server startup).
SQL	Text of the SQL that generated the cache entry.
Business Model	Name of the business model associated with the cache entry.

To view cache information

- Click an individual server in the Explorer pane, and then select View > Cache.

Session Information

The Session view is available for Analytics Servers. The information is arranged in two windows, described in [Table 47](#).

- Session window—Appears on the top. Shows users currently logged on to the Analytics Server.
- Request window—Appears on the bottom. Shows active query requests for the user selected in the Session window.

Table 47 describes the information that appears in the Session window.

Table 47. Session Window Columns (Top Window)

Column	Description
Session ID	Unique internal identifier that the Analytics Server assigns each session when the session is initiated.
User	Name of the user connected.
Client Type	Type of client session. The client type of Administration is reserved for the user logged in with the Siebel Business Analytics administrator user ID.
Catalog	Name of the Presentation layer catalog to which the session is connected.
Repository	Logical name of the repository to which the session is connected.
Logon Time	Timestamp when the session logged on to Analytics Server.
Last Active Time	Timestamp of the last activity on the session or the query.

Table 48 describes the information that appears in the Request window.

Table 48. Request Window Columns (Bottom Window)

Column	Description
Session ID	Unique internal identifier that Analytics Server assigns each session when the session is initiated.
Request ID	Unique internal identifier that Analytics Server assigns each query when the query is initiated.
Start Time	Time of the initial query request.

Table 48. Request Window Columns (Bottom Window)

Column	Description
Last Active Time	Timestamp of the last activity on the session or the query.
Status	<p>These are the possible values. Due to the speed at which some processes complete, you may not see all values for any given request or session.</p> <ul style="list-style-type: none"> ■ Idle—There is presently no activity on the request or session. ■ Fetching—The request is being retrieved. ■ Fetched—The request has been retrieved. ■ Preparing—The request is being prepared for processing. ■ Prepared—The request has been prepared for processing and is ready for execution. ■ Executing—The request is currently running. To kill a request, select it and click the Kill Request button. The user will receive an informational message indicating that the Analytics Server administrator cancelled the request. ■ Executed—The request has finished running. ■ Succeeded—The request ran to completion successfully. ■ Canceled—The request has been canceled. ■ Failed—An error was encountered during the processing or running of the request.

To view session information

- Select a server in the Explorer pane, and then select View > Sessions.

Session information for the server is displayed in the Information pane. It shows all users logged into the server and all current query requests for each user.

To disconnect a session

- In the Session view, right-click the session in the Session window (top window) and click Disconnect.

To kill a query request

- In the Session view, right-click the request in the Request window (bottom window) and click Kill Request.

Server Information

Selecting Server info from the View menu provides information about the cluster server such as server version number.

Managing Clustered Servers

In Status view, the Cluster Manager allows you to manage clustered servers. The following actions are available from the toolbar:

- **Start.** Starts the clustered server. When the server is ready to start accepting sessions, it will transition to Online status.
- **Quiesce.** Quiesces the clustered server. Any queued and in-progress sessions will be allowed to complete, but no new sessions will be assigned to the server. If this action would mean that no servers would be left online, a message will alert you that the cluster will be disabled if you proceed.
- **Stop.** Stops the clustered server. Any queued and in-progress sessions will be stopped. A warning is issued if the server about to be stopped is the one to which the Administration Tool is connected.
- **Close.** Closes the Cluster Manager window.
- **Restart Servers.** Sequentially restarts all the servers except the one to which the Administration Tool is connected. This option is useful if an online repository change has been made and published and you want servers other than the master to restart in order to pick up the changes.

To manage clustered servers

- 1 In the Explorer pane, click the plus sign (+) to the left of the Server icon to display the servers in the cluster.
- 2 In the Information pane, select a server.
- 3 Select Action, and then select one of the available options.

When the operation finishes, the status of the clustered server will be refreshed automatically.

Performance Considerations

This section describes characteristics of the Cluster Server feature that may influence the performance of clustered Analytics Servers. You should consider these points when implementing the Cluster Server feature.

- Sessions are assigned to an Analytics Server when the session is established. A session is assigned to the server with the fewest sessions. As a result, the load on Analytics Servers can vary and Analytics Servers brought online into an operational cluster may not receive work for some time.
- Because each Analytics Server maintains its own local query results cache, back-end databases may receive the same query from multiple Analytics Servers even though the result is cached.
- Because each Analytics Server maintains its own local query results cache, Analytics Servers that are brought online into an operational cluster may respond to queries more slowly while their local cache is being populated.

- Because each Analytics Server has an independent copy of each repository and hence its own back-end connection pools, back-end databases may experience as many as $N \times M$ connections, where N is the number of active servers in the cluster and M is the maximum sessions allowed in the connection pool of a single repository. Therefore, it may be appropriate to reduce the maximum number of sessions configured in session pools.

16 Security in Siebel Business Analytics

The Analytics Server provides secure access control at any level. This section contains the following topics:

- [Analytics Security Manager on page 327](#)
- [Authentication Options on page 336](#)
- [Managing Query Execution Privileges on page 343](#)

Analytics Security Manager

Oracle's Siebel Analytics Security Manager displays all security information for a repository. You can use the Security Manager to configure users and groups, synchronize LDAP users and groups, set access privileges for objects such as tables and columns, set filters on information, and set up a managed query environment in which you have a great deal of control over when users can access data.

NOTE: You should read this section to understand the basics about security and setting up authentication. After reading this section, see details about configuring security for Siebel Business Analytics applications in *Siebel Business Analytics Applications Installation and Administration Guide*.

The Analytics Server and Web client support industry-standard security for login and password encryption. When an end user enters a login and password in the Web browser, the Analytics Server uses the Hyper Text Transport Protocol Secure (HTTPS) standard to send the information to a secure port on the Siebel Business Analytics Web server. From the Siebel Business Analytics Web Server, the information is passed through ODBC to the Analytics Server, using Triple DES (Data Encryption Standard). This provides a high level of security (168 bit), preventing unauthorized users from accessing data or Siebel Business Analytics metadata.

At the database level, administrators can implement database security and authentication. Finally, a proprietary key-based encryption provides security to prevent unauthorized users from accessing the Siebel Business Analytics metadata repository.

This section includes the following topics:

- [Working with Users on page 328](#)
- [Working with Groups on page 329](#)
- [Importing Users and Groups from LDAP on page 333](#)

Working with Users

User accounts can be defined explicitly in an analytics repository or in an external source (such as a database table or an LDAP server). However user accounts are defined, users need to be authenticated by the Analytics Server for a session to take place unless the administrator has configured the system to bypass Analytics Server security. For more information, see [“Bypassing Analytics Server Security” on page 342](#).

Users defined explicitly in a repository can access business models in that repository, but they cannot span repositories.

The Administrator user account is created automatically when a repository is created and cannot be deleted. For more information about the Administrator user account, see [“About the Analytics Server Administrator Account” on page 329](#).

This section includes the following topics:

- [Adding a New User to a Repository on page 328](#)
- [About the Analytics Server Administrator Account on page 329](#)

Adding a New User to a Repository

Use this procedure to add a new user to a repository.

To add a new user to a repository

- 1 Open a repository in the Administration Tool.
- 2 Display the security manager by selecting Manage > Security.
- 3 Select Action > New > User to open the User dialog box.
- 4 Type a name and password for the user.
- 5 If you want to log queries for this user in the query log, change the query logging level to 1 or 2. For more information about query logging, see [“Setting a Logging Level” on page 243](#).
- 6 Click OK.

This creates a new user with default rights granted to it. In the NQSCfg.INI file, the default rights are specified by the entry DEFAULT_PRIVILEGES.

- 7 To modify the user's permissions, open the User dialog by double-clicking on the user icon you want to modify. If you click Permissions, you can change permissions for multiple columns.
- 8 Specify the password expiration option.
 - If the user's password should never expire, select the option Password Never Expires.
 - If you want the user's password to expire, use the Days drop-down list to select the number of days to elapse before the user's password will expire. The maximum interval is 365 days.

When the specified number of days passes after the password is created or changed, the password expires and must be changed.

- 9 You can grant rights to the user individually, through groups, or a combination of the two. To grant membership in a group, check as many groups as you want the user to be a part of in the Group Membership portion of the dialog box.
- 10 To specify specific database logon IDs for one or more databases, type the appropriate user IDs and passwords for the user in the Logons tab of the User dialog box.
NOTE: If a user specifies database-specific logon IDs in the DSN used to connect to the Analytics Server, the logon IDs in the DSN are used if the administrator has configured a connection pool with no default database-specific logon ID and password. For information about configuring the connection pools to support database-specific logon IDs, see [“Creating or Changing Connection Pools” on page 68](#).
- 11 Set up any query permissions for the user. For information, see [“Managing Query Execution Privileges” on page 343](#).

About the Analytics Server Administrator Account

The Analytics Server Administrator account (user ID of Administrator) is a default user account in every analytics repository. This is a permanent account. It cannot be deleted or modified other than to change the password and logging level. It is designed to perform all administrative tasks in a repository, such as importing physical schemas, creating business models, and creating users and groups.

NOTE: The Analytics Server Administrator account is not the same as the Windows NT and Windows 2000 Administrator account. The administrative privileges granted to this account function only within the Analytics Server environment.

When you create a new repository, the Administrator account is created automatically and has no password assigned to it. You should assign a password for the Administrator account as soon as you create the repository. The Administrator account created during the installation of the analytics repository, that is, the repository shipped with Siebel Business Analytics, has the default password SADMIN.

The Administrator account belongs to the Administrators group by default and cannot be deleted from it. The person logged on using the Administrator user ID or any member of the Administrators group has permissions to change anything in the repository. Any query issued from the Administrator account has complete access to the data; no restrictions apply to any objects.

NOTE: You can set the minimum length for passwords in the NQSCfg.ini file using the MINIMUM_PASSWORD_LENGTH setting.

Working with Groups

The Analytics Server allows you to create *groups* and then grant membership in them to users or other groups.

You can think of a group as a set of security attributes. The Analytics Server groups are similar to groups in Windows NT and Windows 2000, and to groups or roles in database management systems (DBMS). Like Windows NT and Windows 2000, and database groups or roles, Analytics Server groups can allow access to objects. Additionally, Analytics Server groups can explicitly deny particular security attributes to its members.

Groups can simplify administration of large numbers of users. You can grant or deny sets of privileges to a group and then assign membership in that group to individual users. Any subsequent modifications to that group will affect all users who belong to it. Externally defined users can be granted group membership by use of the GROUP session variable. For more information about session variables, see [“Using System Session Variables” on page 300](#).

This section includes the following topics:

- [Predefined Administrators Group on page 330](#)
- [Defined Groups on page 330](#)
- [Group Inheritance on page 331](#)
- [Adding a New Group on page 332](#)
- [Viewing Member Hierarchies on page 333](#)

Predefined Administrators Group

The Analytics Server has one predefined group, the Analytics Server Administrators group. Members of this group have the authority to access and modify any object in a repository. The predefined Analytics Server Administrator user ID is automatically a member of the Analytics Server Administrators group.

Use caution in granting membership in the Administrators group to users or other groups. Membership in the Administrators group supersedes all privileges granted to a user, either through groups or explicitly through the user privileges. Any user who is a member of the Administrators group has all of the privileges of the Administrator user.

Defined Groups

You can create an unlimited number of groups in an analytics repository. Each group can contain explicitly granted privileges or privileges granted implicitly using membership in another group. For more information about setting up a group, see [“Adding a New Group” on page 332](#).

For example, you can create one group that denies access to the repository on Mondays and Wednesdays (Group1), another group that denies access on Saturdays and Sundays (Group2), and another that denies access on Tuesdays, Thursdays, and Fridays (Group3). Users who are members of Group2 can access the system only during weekdays, users who are members of Group1 and Group3 can access the system only on weekends, and so on.

Group Inheritance

Users can have explicitly granted privileges. They can also have privileges granted through membership in groups, that in turn can have privileges granted through membership in other groups, and so on. Privileges granted explicitly to a user have precedence over privileges granted through groups, and privileges granted explicitly to the group take precedence over any privileges granted through other groups.

If there are multiple groups acting on a user or group *at the same level* with conflicting security attributes, the user or group is granted the least restrictive security attribute. Any explicit permissions acting on a user take precedence over any privileges on the same objects granted to that user through groups.

Example 1

Suppose you have a user (User1) who is explicitly granted permission to read a given table (TableA). Suppose also that User1 is a member of Group1, that explicitly denies access to TableA. The resultant privilege for User1 is to read TableA, as shown in [Figure 27](#).

Because privileges granted directly to the user take precedence over those granted through groups, User1 has the privilege to read TableA.

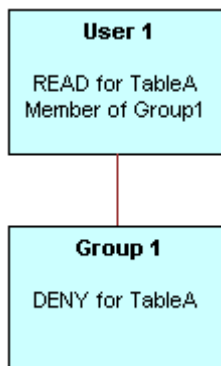


Figure 27. User Privileges and Group Privileges

Example 2

Consider the situation shown in [Figure 28](#).

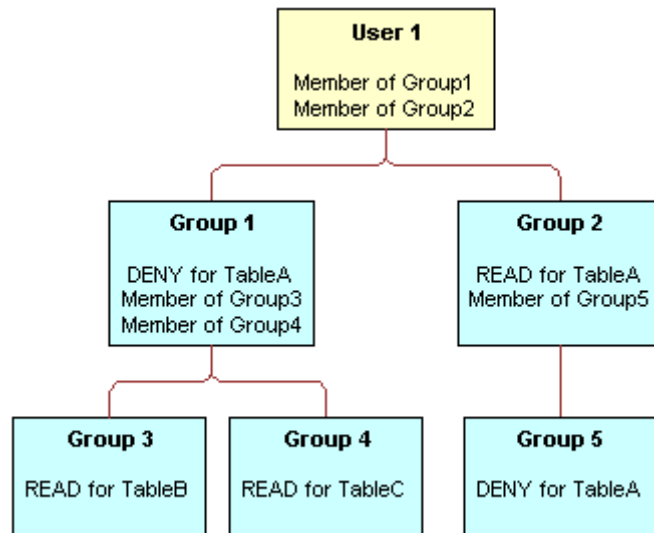


Figure 28. Privileges Example

These are the resulting privileges:

- User1 is a direct member of Group1 and Group2, and is an indirect member of Group3, Group4, and Group5.
- Because Group5 is at a lower level of precedence than Group2, its denial of access to TableA is overridden by the READ privilege granted through Group2. The result is that Group2 provides READ privilege on TableA.
- The resultant privileges from Group1 are DENY for TableA, READ for TableB, and READ for TableC.
- Because Group1 and Group2 have the same level of precedence and because the privileges in each cancel the other out (Group1 denies access to TableA, Group2 allows access to TableA), the less restrictive level is inherited by User1; that is, User1 has READ access to TableA.
- The total privileges granted to User1 are READ access for TableA, TableB, and TableC.

Adding a New Group

The following procedure explains how to add a new group to a repository.

To add a new group to a repository

- 1 Open a repository in the Administration Tool. (The repository can be opened in either online or offline mode.)
- 2 Display the security window by selecting Manage > Security.

- 3 Select Action > New > Group from menu.

The Group dialog box appears.

NOTE: You can also select the Group icon in the left pane, and then right-click on white space in the left pane and select New Security Group from the right-click menu.

- 4 Type a name for the group and click OK.

This creates a new group with no rights granted to it.

- 5 To modify the group's permissions, open the Group dialog by double-clicking on the group icon you want to modify. If you click on Permissions, you can change permissions for multiple columns.

- 6 You can grant rights to the group by adding other groups, by explicit configuration for the group, or a combination of the two. To grant membership to a group, click Add and select any users or groups you want to grant membership. Click OK after you have selected the groups and users.

- 7 Set up any query permissions for the group. For information, see ["Managing Query Execution Privileges" on page 343](#).

NOTE: Unlike the User dialog box, the Group dialog box does not allow you to select a logging level. The logging level is a user attribute and cannot be specified for a group.

Viewing Member Hierarchies

Use the following procedures to view member hierarchies.

To view member hierarchies in the Security Manager

- Click the hierarchy icon in the left pane of the Security Manager, and then expand the tree in the right pane.

To view member hierarchies in the Query Repository dialog box

- 1 Select Tools > Query Repository from the main menu of the Administration Tool.
- 2 To see all groups, select Security Groups from the Type drop-down list and click Query.
- 3 To see all users, select Users from the Type drop-down and click Query.
- 4 To see what groups a group is a member of, select the group and click Parent. For example, to see what groups Group1 is a member of, select Group1 and click the Parent button.

Importing Users and Groups from LDAP

If your organization uses Lightweight Directory Access Protocol (LDAP), you can import your existing LDAP users and groups to a repository. After imported, all normal Analytics Server user and group functions are available. You can resynchronize your imported list at any time.

You can also authenticate against LDAP as an external source. When you do this, users are not imported into the repository. Users are authenticated, and their group privileges determined, when they log on. For more information about using LDAP authentication, see [“Setting Up LDAP Authentication” on page 337](#).

This section includes the following topics:

- [Configuring an LDAP Server on page 334](#)
- [Importing Users from LDAP on page 336](#)
- [Synchronizing Users and Groups with LDAP on page 336](#)

NOTE: If a user exists in both the repository and in LDAP, the local repository user definition takes precedence. This allows the Analytics Server Administrator to reliably override users that exist in an external security system.

Configuring an LDAP Server

This section explains how to configure LDAP authentication for the repository.

NOTE: For information about the basics of security and setting up authentication, see [“Analytics Security Manager” on page 327](#).

For instances of Siebel Business Analytics that use ADSI as the authentication method, the following AD configuration options should be used when configuring the AD instance:

- In Log On To, check All Computers or, if you list some computers, include the AD server as a Logon workstation.
- The following option must not be checked:
User must change password at next logon

In the Administration Tool, the CN user used for the BIND DN of the LDAP Server section must have both ldap_bind and ldap_search authority.

NOTE: The Analytics Server uses clear text passwords in LDAP authentication. Make sure your LDAP Servers are set up to allow this.

To configure LDAP authentication for the repository

- 1 Open a repository in the Administration Tool in offline or online mode.
- 2 From the application menu, choose Manage > Security.
- 3 From the Security Manager menu, choose Action > New > LDAP Server.
- 4 In the LDAP Server dialog box, in the General tab, complete the necessary fields. The following list of fields (or buttons) and descriptions contain additional information to help you set up the LDAP server:
 - **Host name.** The name of your LDAP server.
 - **Port number.** The default LDAP port is 389.
 - **LDAP version.** LDAP 2 or LDAP 3 (versions). The default is LDAP 3.

- **Base DN.** The base distinguished name (DN) identifies the starting point of the authentication search. For example, if you want to search all of the entries under the o=Siebel . com subtree of the directory, o=Siebel . com is the base DN.

- **Bind DN and Bind Password.** The optional DN and its associated user password that are required to bind to the LDAP server.

If these two entries are blank, *anonymous binding* is assumed. For security reasons, not all LDAP servers allow anonymous binding.

These fields are optional for LDAP V3, but required for LDAP V2, because LDAP V2 does not support anonymous binding.

These fields are required if you select the ADSI check box. If you leave these fields blank, a warning message appears asking if you want to leave the password empty anyway. If you click Yes, anonymous binding is assumed.

- **Test Connection.** Use this button to verify your parameters by testing the connection to the LDAP server.

- 5 Click the Advanced tab, and type the required information. The following list of fields and descriptions contain additional information to help you set up the LDAP server:

NOTE: The Analytics Server maintains an authentication cache in memory that improves performance when using LDAP to authenticate large numbers of users. Disabling the authentication cache can slow performance when hundreds of sessions are being authenticated.

- **Connection timeout.** When the Administration Tool attempts to connect to an LDAP server for import purposes or the Analytics Server attempts to connect to an LDAP server for user authentication, the connection will time out after the specified interval.

- **Domain identifier.** Typically, the identifier is a single word that uniquely identifies the domain for which the LDAP object is responsible. This is especially useful when you use multiple LDAP objects. If two different users have the same user ID and each is on a different LDAP server, you can designate domain identifiers to differentiate between them. The users log in to the Analytics Server using the following format:

domai n_i d/user_i d

If a user enters a user id without the domain identifier, it will be authenticated against all available LDAP servers in turn. If there are multiple users with the same ID, only one user can be authenticated.

- **ADSI.** (Active Directory Service Interfaces) A type of LDAP server. If you select the ADSI check box, Bind DN and Bind password are required.
- **SSL.** (Single Socket Layer) Check this box to enable this.
- **User Name Attribute Type.** This uniquely identifies a user. In many cases, this is the RDN (relative distinguished name). Typically, you accept the default value. For most LDAP servers, you would use the user ID. For ADSI, use sAMAccountName.

NOTE: Cache settings and SSL key database can be configured in Oracle's Siebel Tools > Options > Repository. For more information, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Importing Users from LDAP

You can import selected users or groups, or you can import all users or groups. If you have previously performed an import, you can choose to synchronize the repository with the LDAP server.

To import LDAP users and groups to a repository

- 1 Open a repository in the Administration Tool in offline or online mode.
- 2 From the application menu, choose Manage > Security.
- 3 In the Security Manager, select LDAP Servers in the left pane to display the configured LDAP servers in the right pane. Select the LDAP server from which you want to import users or groups, and select Import... from the right-click menu. (You can also select the server and then select LDAP > Import.)

You can choose to import selected users or groups, or you can import all users and groups. If you have previously done an import, you can choose to synchronize the repository with the LDAP server.

- 4 Select the users you want to import and click Import.

You can import groups by selecting Groups from the drop down list instead of Users.

Synchronizing Users and Groups with LDAP

You can refresh the repository users and groups with the current users and groups on your LDAP server. After selecting the appropriate LDAP server, select LDAP > Synchronize (or choose Synchronize from the right-click menu).

Synchronization updates your list of repository users and groups to mirror your current LDAP users and groups. Users and groups that do not exist on your LDAP server are removed from the repository. The special user Administrator and the special group Administrators always remain in your repository and are never removed.

Properties of users already included in the repository are not changed by synchronization. If you have recycled a login name for another user, drop that name from your repository prior to synchronization. This assures that the process will import the new LDAP user definition.

NOTE: With external LDAP authentication (discussed in the next section), import and synchronization are not really necessary. The primary use for import is to make it easy to copy LDAP users as Siebel Business Analytics users for testing.

Authentication Options

Authentication is the process by which a system uses a user ID and password to verify that a user has the necessary permissions and authorizations to log in and access data. The Analytics Server authenticates each connection request it receives.

The Analytics Server supports the following authentication types:

- [Setting Up LDAP Authentication on page 337](#)

- [Setting Up External Table Authentication on page 338](#)
- [Setting Up Database Authentication on page 340](#)
- [About Delivers and Database Authentication on page 340](#)
- [Maintaining Analytics Server Internal Authentication on page 341](#)

Setting Up LDAP Authentication

Instead of storing user IDs and passwords in an analytics repository, you can set up the Analytics Server to pass the user ID and password typed by the user to an LDAP server for authentication. The server uses clear text passwords in LDAP authentication. Make sure your LDAP servers are set up to allow this.

In addition to basic user authentication, the LDAP server can also provide the Analytics Server with other information, such as the user display name (used by Siebel Business Analytics Web) and the name of any groups to which the user belongs. The LDAP server can also provide the names of specific database catalogs or schemas to use for each user when querying data. This information is contained in LDAP variables that get passed to Siebel Business Analytics *session variables* during the process of user authentication. For more information about session variables, see [“About Session Variables” on page 299](#).

LDAP authentication uses Siebel Business Analytics session variables, that you define using the Variable Manager of the Administration Tool. For more information about the Variable Manager, see [“Using the Variable Manager” on page 297](#).

Session variables get their values when a user begins a session by logging on. Certain session variables, called *system session variables*, have special uses. The variable USER is a system variable that is used with LDAP authentication. For more information about the USER system variable, see [“Using System Session Variables” on page 300](#).

To set up LDAP authentication, you define a system variable called USER and associate it with an LDAP initialization block that is associated with an LDAP server. Whenever a user logs into the Analytics Server, the user ID and password will be passed to the LDAP server for authentication. After the user is authenticated successfully, other session variables for the user could also be populated from information returned by the LDAP server.

The following discussion assumes that an LDAP initialization block has already been defined. Setting up an LDAP initialization block is explained in [“Configuring an LDAP Server” on page 334](#).

NOTE: The presence of a defined session system variable USER determines that external authentication is done for users not defined in the repository. Associating USER with an LDAP initialization block determines that the user will be authenticated by LDAP. To provide other forms of authentication, associate the USER variable with an initialization block associated with an external database or XML source. For more information, see [“Setting Up External Table Authentication” on page 338](#).

To define the USER session system variable for LDAP authentication

- 1 Select Manage > Variables from the Administration Tool menu.

- 2 Select the System leaf of the tree in the left pane.
- 3 Right-click on the right pane and select New USER.
- 4 In the Session Variable - USER dialog box, select the appropriate LDAP initialization block from the Initialization Block drop-down list.
The selected initialization block provides the USER session system variable with its value.
- 5 Click OK to create the USER variable.

Setting the Logging Level

Use the system variable LOGLEVEL to set the logging level for users who are authenticated by an LDAP server. See [“Setting a Logging Level” on page 243](#) for more information.

Setting Up External Table Authentication

Instead of storing user IDs and passwords in an analytics repository, you can maintain lists of users and their passwords in an external database table and use this table for authentication purposes. The external database table contains user IDs and passwords, and could contain other information, including group membership and display names used for Siebel Business Analytics Web users. The table could also contain the names of specific database catalogs or schemas to use for each user when querying data.

NOTE: If a user belongs to multiple groups, the group names should be included in the same column separated by semicolons.

External table authentication can be used in conjunction with database authentication. If external table authentication succeeds, then database authentication is not performed. If external table authentication fails, then database authentication is performed.

See [“Setting Up Database Authentication” on page 340](#), and [“Order of Authentication” on page 342](#) for additional details.

External table authentication uses Siebel Business Analytics session variables that you define using the Variable Manager of the Administration Tool. For more information about the Variable Manager, see [“Using the Variable Manager” on page 297](#).

Session variables get their values when a user begins a session by logging on. Certain session variables, called *system variables*, have special uses. The variable USER is a system variable that is used with external table authentication.

To set up external table authentication, you define a system variable called USER and associate it with an *initialization block* that is associated with an external database table. Whenever a user logs in, the user ID and password will be authenticated using SQL that queries this database table for authentication. After the user is authenticated successfully, other session variables for the user could also be populated from the results of this SQL query. For more information about session variables, see [“About Session Variables” on page 299](#).

The presence of a defined system variable USER determines that external authentication is done. Associating USER with an external database table initialization block determines that the user will be authenticated using the information in this table. To provide other forms of authentication, associate the USER system variable with an initialization block associated with a LDAP server or XML source. For more information, see [“Setting Up LDAP Authentication” on page 337](#).

To set up external table authentication

- 1 Import information about the external table into the Physical layer. In this illustration, the database sql_nqsecurity contains a table named securitylogons and has a connection pool named External Table Security.
- 2 Select Manage > Variables to open the Variable Manager.
- 3 Select Initialization Blocks on the left tree pane.
- 4 Right-click on white space in the right pane, and then click on New Initialization Block from the right-click menu.
- 5 In the Initialization Block dialog box, type the name for the initialization block.
- 6 Select Database from the Data Source Connection drop-down list.
- 7 Click Browse to search for the name of the connection pool this block will use.
- 8 In the Initialization String area, type the SQL statement that will be issued at authentication time.

The values returned by the database in the columns in your SQL will be assigned to variables. The order of the variables and the order of the columns will determine which columns are assigned to which variables. Consider the SQL in the following example:

```
select username, grp_name, SalesRep, 2 from securitylogons where username =
':USER' and pwd = ':PASSWORD'
```

This SQL contains two constraints in the WHERE clause:

- :USER (note the colon) equals the ID the user entered when logging on.
- :PASSWORD (note the colon again) equals the password the user typed.

The query will return data only if the user ID and password match values found in the specified table.

You should test the SQL statement outside of the Analytics Server, substituting valid values for :USER and :PASSWORD to verify that a row of data returns.

- 9 If this query returns data, the user is authenticated and session variables will be populated. Because this query returns four columns, four session variables will be populated. Create these variables (USER, GROUP, DISPLAYNAME, and LOGLEVEL) by clicking New in the dialog's Variables tab.

If a variable is not in the desired order, click on the variable you want to reorder and use the Up and Down buttons to move it.
- 10 Click OK to save the initialization block.

Setting Up Database Authentication

The Analytics Server can authenticate users through database logons. If a user has read permission on a specified database, the user will be trusted by the Analytics Server. Unlike operating system authentication, this authentication can be applied to Siebel Business Analytics Web users. For information, see [“About Delivers and Database Authentication” on page 340](#).

Database authentication can be used in conjunction with external table authentication. If external table authentication succeeds, then database authentication is not performed. If external table authentication fails, then database authentication is performed.

See [“Setting Up External Table Authentication” on page 338](#) and [“Order of Authentication” on page 342](#) for additional details.

Database authentication requires the user ID to be stored in the analytics repository.

To set up database authentication

- 1 Create users in the repository named identically to the users in a database. Passwords are not stored in the repository.
- 2 Assign the permissions (including group memberships, if any) you want the users to have.
- 3 Specify the authentication database in the Security section of the NQSCfg.INI file. (See *Siebel Business Analytics Platform Installation and Configuration Guide* for additional information.)
- 4 Create a DSN for the database.
- 5 Import the database into the Physical layer. You do not need to import the physical table objects. The database name in the Physical layer has to match the database name in the NQSCfg.INI file (as specified in Step 3).
- 6 Set up the connection pool without a shared logon.

When a user attempts to log on to the Analytics Server, the server attempts to use the logon name and password to connect to the authentication database, using the first connection pool associated with it. If this connection succeeds, the user is considered to be authenticated successfully.

If the logon is denied, the Analytics Server issues a message to the user indicating an invalid user ID or password.

About Delivers and Database Authentication

In Siebel Business Analytics applications, users are always created in the operational application database, never in the analytics repository. The analytics repository is preconfigured for database authentication.

Siebel Scheduler Server runs Delivers jobs for users without accessing or storing their passwords. Using a process called impersonation, the Scheduler uses one user ID and password with administrator privileges that can act on behalf of other users. The Scheduler initiates an iBot by logging on to Siebel Business Analytics Web with that administrator ID and password.

For Delivers to work, all database authentication must be performed in only one connection pool, and that connection pool can only be selected in an initialization block for the USER system session variable. This is typically called the Authentication Initialization Block. When impersonation is used, this initialization block is skipped. All other initialization blocks must use connection pools that do not use database authentication.

CAUTION: Using an authentication initialization block is the only initialization block in which it is acceptable to use a connection pool in which :USER and :PASSWORD are passed to a physical database.

For other initialization blocks, SQL statements can use :USER AND :PASSWORD. However, because Oracle's Siebel Scheduler Server does not store user passwords, the WHERE clause must be constructed as shown in the following example:

```
select username, groupname, dbname, schemaname from users
where username=:USER'
NQS_PASSWORD_CLAUSE(and pwd=:PASSWORD')NQS_PASSWORD_CLAUSE
```

NOTE: When impersonation is used, everything in the parentheses is extracted from the SQL statement at runtime.

For more information, see Oracle's Siebel Delivers example in [“Examples of Initialization String SQL Statements” on page 309](#).

Maintaining Analytics Server Internal Authentication

You can maintain lists of users and their passwords in the analytics repository using the Administration Tool. The Analytics Server will attempt to authenticate users against this list when they log on unless another authentication method has already succeeded, or database authentication has been specified in the NQSConfig.INI file.

See [“Order of Authentication” on page 342](#) for additional information.

The Analytics Server user IDs are stored in nonencrypted form in an analytics repository and are caseinsensitive. Passwords are stored in encrypted form and are casesensitive. The Analytics Server user IDs can be used to access any business model in a repository provided that the users have the necessary access privileges. User IDs are valid only for the repository in which they are set up. They do not span multiple repositories.

NOTE: If you are using LDAP or external table authentication, passwords are not stored in the analytics repository.

Changing Analytics Server User Passwords

You can change user passwords in the Administration Tool.

To change a user password

- 1 Select Manage > Security.
- 2 In the Security Manager dialog box, select Users in the left pane.

- 3 In the right pane, right-click the user whose password you want to change.
- 4 Select Properties from the shortcut menu.
- 5 In the User tab, type the new password.
- 6 In the Confirm Password text box, type the password again, and then click OK.

Order of Authentication

If the user does not type a logon name, then OS authentication is triggered, unless OS authentication is explicitly turned off in the NQSSConfig.INI file. For more information, see *Siebel Business Analytics Platform Installation and Configuration Guide*. Additionally, OS authentication is not used for Siebel Business Analytics Web users.

The Analytics Server populates session variables using the initialization blocks in the desired order that are specified by the dependency rules defined in the initialization blocks. If the server finds the session variable USER, it performs authentication against an LDAP server or an external database table, depending on the configuration of the initialization block with which the USER variable is associated.

Analytics Server internal authentication (or, optionally, database authentication) occurs only after these other possibilities have been considered.

Bypassing Analytics Server Security

Another option is to bypass Analytics Server security and rely on the security provided by issuing user-specific database logons and passwords when the Analytics Server submits queries to databases. The databases can then determine whether the query will be performed for the user.

The Analytics Server issues queries to databases in one of the following ways:

- By using the user IDs and passwords configured in connection pools when the connection pool property Shared Login has been checked.
- With database-specific user IDs and passwords that are specific to each user. Configure the database user IDs and passwords in the user's profile in the analytics repository.
- If you are using database-specific login information, connection pooling needs to be set up without the Shared Login property, allowing it to accept database-specific user IDs and passwords.

For more information about connection pools, see ["Setting Up Connection Pools" on page 65](#). For more information about the NQSSConfig.ini file, see *Siebel Business Analytics Platform Installation and Configuration Guide*.

Bypass Analytics Server security by setting the authentication type in the NQSSConfig.INI file:

```
AUTHENTICATION_TYPE = BYPASS_NQS;
```

Managing Query Execution Privileges

The Analytics Server allows you to exercise varying degrees of control over the repository information that a user can access.

Controlling query privileges allows you to manage the query environment. You can put a high level of query controls on users, no controls, or somewhere in between. The following list contains some types of activities you may want to limit:

- Restricting query access to specific objects, including rows and columns, or time periods
 - **Objects.** If you explicitly deny access to an object that has child objects, the user will be denied access to the child objects. For example, if you explicitly deny access to a particular physical database object, you are implicitly denying access to all of the physical tables and physical columns in that catalog.

If a user or group is granted or disallowed privileges on an object from multiple sources (for example, explicitly and through one or more groups), the privileges are used based on the order of precedence, as described in [“Group Inheritance” on page 331](#).

You can grant or disallow the ability to execute direct database requests for a user or group.
 - **Time periods.** If you do not select a time period, access rights remain unchanged. If you allow or disallow access explicitly in one or more groups, the user is granted the least restrictive access for the defined time periods. For example, suppose a user is explicitly allowed access all day on Mondays, but belongs to a group that is disallowed access during all hours of every day. This means that the user will have access on Mondays only.
- Controlling runaway queries by limiting queries to a specific number of rows or maximum run time
- Limit queries by setting up filters for an object

All restrictions and controls can be applied at the user level, at the group level, or a combination of the two.

To limit queries by objects for a user or group

- 1 From the Administration Tool menu bar, choose Manage > Security.
- 2 In the Security Manager dialog box, in the tree pane, select Users or Groups.
- 3 In the right pane, right-click the name that you want to change and select Properties.
- 4 In the User or Group dialog box, click Permissions.
- 5 In the User/Group Permissions dialog box, click the General tab and perform the following steps:
 - a In the General tab, to explicitly allow or disallow access to one or more objects in the repository, click Add.
 - b In the Browse dialog box, in the Name list, select the objects you want to change, and then click Select.

- c In the User/Group Permissions dialog box, assign the permissions by selecting or clearing the Read check box for each object.

(Default is a check) If the check box contains a check, the user has read privileges on the object. If the check box contains an X, the user is disallowed read privileges on the object. If it is blank, any existing privileges (for example, through a group) on the object apply.

For more information about assigning permissions, see [“Setting Permissions for Repository Objects” on page 35](#).

- 6 To explicitly allow or disallow populate privilege or the ability to execute direct database requests for specific database objects, perform the following steps:
 - a Click the Query Limits tab and select the database.
 - b In the Populate Privilege drop-down list, select Allow or Disallow.

NOTE: For the selected user or group, this overrides the database property Allow populate queries for all.
 - c To explicitly allow or disallow the ability to execute direct database requests for specific database objects, in the Execute Direct Database Requests drop-down list, select Allow or Disallow.

NOTE: For the selected user or group, this overrides the database property Allow direct database requests for all.
- 7 Click OK twice to return to the Security Manager dialog box.

To limit queries by number of rows received by a user or group

- 1 From the Administration Tool menu bar, choose Manage > Security.
- 2 In the Security Manager dialog box, in the tree pane, select Users or Groups.
- 3 In the right pane, right-click the name that you want to change and select Properties.
- 4 In the User or Group dialog box, click the Permissions tab.
- 5 In the User/Group Permissions dialog box, click the Query Limits tab and expand the dialog box to see all columns.
- 6 To specify or change the maximum number of rows each query can retrieve from a database, in the Query Limits tab, perform the following steps:
 - a In the Max Rows column, type the maximum number of rows.
 - b In the Status Max Rows field, select a status using [Table 49](#) as a guide.
- 7 Click OK twice to return to the Security Manager dialog box.

To limit queries by maximum run time or to time periods for a user or group

- 1 From the Administration Tool menu bar, choose Manage > Security.
- 2 In the Security Manager dialog box, in the tree pane, select Users or Groups.
- 3 In the right pane, right-click the name that you want to change and select Properties.

- 4 In the User or Group dialog box, click the Permissions tab.
- 5 In the User/Group Permissions dialog box, click the Query Limits tab and expand the dialog box to see all columns.
- 6 To specify the maximum time a query can run on a database, in the Query Limits tab, perform the following steps:
 - a In the Max Time column, select the number of minutes.
 - b From the Status Max Time drop-down list, select a status using [Table 49](#) as a guide.
- 7 To restrict access to a database during particular time periods, in the Restrict column, click the ellipsis button.
- 8 In the Restrictions dialog box, perform the following steps:
 - a To select a time period, click the start time and drag to the end time.
 - b To explicitly allow access, click Allow.
 - c To explicitly disallow access, click Disallow.
- 9 Click OK twice to return to the Security Manager dialog box.

To limit queries by setting up a filter on an object for a user or group

- 1 From the Administration Tool menu bar, choose Manage > Security.
- 2 In the Security Manager dialog box, in the tree pane, select Users or Groups.
- 3 In the right pane, right-click the name that you want to change and select Properties.
- 4 In the User or Group dialog box, click Permissions.
- 5 In the User/Group Permissions dialog box, click the Filters tab.
- 6 In the Filters tab, to add an object to filter, perform the following steps:
 - a Click Add.
 - b In the Browse dialog box, in the Names list, locate and double-click the object on which you want to filter.
 - c Select the object and click Select.
- 7 In the User/Group Permissions Filters dialog box, perform the following steps:
 - a Scroll to the right to see the Business Model Filter column.
 - b Click the Business Model Filter ellipsis button for the selected object.
- 8 In the Expression Builder dialog box, create a logical filter, and then click OK.
- 9 In the User/Group Permissions Filters dialog box, from the Status drop-down list, select a status using [Table 49](#) as a guide.

10 Click OK twice to return to the Security Manager dialog box.

Table 49. Query Privileges Status Fields

Status	Description
Disable	<ul style="list-style-type: none"> ■ Status Max Rows or Status Max Time. When selected, disables any limits set in the Max Rows or Max Time fields. ■ Filter. The filter is not used and no other filters applied to the object at higher levels of precedence (for example, through a group) are used.
Enable	<ul style="list-style-type: none"> ■ Status Max Rows or Status Max Time. This limits the number of rows or time to the value specified. If the number of rows exceeds the Max Rows value, the query is terminated. ■ Filter. The filter is applied to any query that accesses the object.
Ignore	<ul style="list-style-type: none"> ■ Status Max Rows or Status Max Time. Limits will be inherited from the parent group. If there is no row limit to inherit, no limit is enforced. ■ Filter. The filter is not in use, but any other filters applied to the object (for example, through a group) are used. If no other filters are enabled, no filtering will occur.

Assigning Populate Privilege to a User or Group

When a criteria block is cached, the Populate Stored procedure writes the Cache/Saved Result Set value to the database.

NOTE: Any Marketing user who writes a cache entry or saves a result set needs to be assigned the POPULATE privilege for the target database. All Marketing segmentation users and groups need to be assigned this privilege. Typically, all Marketing users are associated with a group and this group is granted the privilege. For more information about marketing cache, see the topic about setting up cache for target levels in *Siebel Marketing Installation and Administration Guide*.

To assign Populate privilege to a user or group

- 1 From the Administration Tool menu bar, choose Manage > Security.
- 2 In the Security Manager dialog box, in the tree pane, select Users or Groups.
- 3 In the right pane, right-click the name that you want to change and select Properties.
- 4 In the User or Group dialog box, click Permissions.
- 5 In the User/Group Permissions dialog box, select the Query Limits tab.
- 6 In the Query Limits list, expand the dialog box to see all columns.
- 7 From the Populate Privilege drop-down list, select Allow or Disallow.

NOTE: For all Marketing data warehouses, set Populate Privilege to Allow.

- 8 Click OK twice to return to the Security Manager dialog box.

17

Using XML as a Data Source for Siebel Business Analytics

This section describes the use of the Extensible Markup Language (XML) as a data source. XML is the universal format for structured documents and data on the Web. It can also be used as a database to store structured data.

The Analytics Server supports various XML access modes, including access through the Analytics Server XML Gateway and its extension, the Data Mining Adapter; and access through an XML ODBC driver.

This section includes the following topics:

- [Locating the XML URL on page 347](#)
- [Using the Analytics Server XML Gateway on page 348](#)
- [Using XML ODBC on page 361](#)
- [XML Examples on page 362](#)

Locating the XML URL

The Analytics Server supports the use of XML data as a data source for the Physical layer in the repository. Depending on the method used to access XML data sources, a data source may be represented by a URL pointing to one of the following sources.

- A static XML file or HTML file that contains XML data islands on the Internet (including intranet or extranet). For example,
tap://216.217.17.176/[DE0A48DE-1C3E-11D4-97C9-00105AA70303].XML
- Dynamic XML generated from a server site. For example,
tap://www.aspserver.com/example.asp
- An XML file or HTML file that contains XML data islands on a local or network drive. For example,
d:/xmldir/example.xml
d:/htmlmdir/island.htm

You can also specify a directory path for local or network XML files, or you can use the asterisk (*) as a wildcard with the filenames. If you specify a directory path without a filename specification (like d:/xmldir), all files with the XML suffix are imported. For example,

d:/xmldir/
d:/xmldir/exam*.xml
d:/htmlmdir/exam*.htm
d:/htmlmdir/exam*.html

- An HTML file that contains tables, defined by a pair of `<table>` and `</table>` tags. The HTML file may reside on the Internet (including intranet or extranet) or on a local or network drive. See ["Accessing HTML Tables" on page 356](#) for more information.

URLs may include repository or session variables, providing support for HTTP data sources that accept user IDs and passwords embedded in the URL; for example: `http://somewebserver/cgi.pl?userid=typeof(session_variable1)&password= typeof(session_variable2)`. (This functionality also allows the Analytics Server administrator to create an XML data source with a location that is dynamically determined by some runtime parameters.) For more information about variables, see [Chapter 14, "Using Variables in the Analytics Repository."](#)

The Analytics Server also supports the use of XSL transformation files (XSLT) or XPath expressions for transforming the XML files or XML data islands in an HTML page.

XSLT is a generalized form of the Cascaded Style Sheet (CSS) for HTML documents as applied to XML documents or text fragments. XPath is a simplified version of XSLT that may be expressed in a one-line statement. For example, `//xml` is an XPath expression instructing the XML processor to extract all elements under the root element `xml`. An XSLT file can also contain an XPath expressions. See <http://www.w3.org/TR/xslt.html> or <http://www.w3.org/TR/xpath> for additional information about XSLT and XPath standards.

NOTE: If the Analytics Server needs to access any nonlocal files (network files or files on the Internet, for example), you need to run the Analytics Server using a valid user ID and password with sufficient network privileges to access these remote files. In Windows NT and Windows 2000, this user ID also needs to have Windows Administrator privileges on the local machine. To change the account under which the server runs, follow the steps described in ["Changing the User ID in Which the Analytics Server Runs"](#) on page 239.

Using the Analytics Server XML Gateway

Using the Analytics Server XML Gateway, the metadata import process flattens the XML document to a tabular form using the stem of the XML filename (that is, the filename less the suffix) as the table name and the second level element in the XML document as the row delimiter. All leaf nodes are imported as columns belonging to the table. The hierarchical access path to leaf nodes is also imported.

The Analytics Server XML Gateway uses the metadata information contained in an XML schema. The XML schema is contained within the XML document or is referenced within the root element of the XML document. Support is currently available for the version of XML schema defined by Microsoft and implemented in its Internet Explorer 5 family of browsers.

Where there is no schema available, all XML data is imported as text data. In building the repository, you may alter the data types of the columns in the Physical layer, overriding the data types for the corresponding columns defined in the schema. The gateway will convert the incoming data to the desired type as specified in the Physical layer. You can also map the text data type to other data types in the Business Model and Mapping layer of the Administration Tool, using the CAST operator.

At this time, the Analytics Server XML Gateway does not support:

- Resolution of external references contained in an XML document (other than a reference to an external XML schema, as demonstrated in the example file in the section ["Analytics Server XML Gateway Example"](#) on page 350).
- Element and attribute inheritance contained within the Microsoft XML schema.

- Element types of a mixed content model (such as XML elements that contain a mixture of elements and CDATA, such as `<p> hello Joe, how are you doing?</p>`).

NOTE: The Analytics Server XML Gateway includes a Data Mining Adapter feature. It allows you to access data sources by calling an executable file or DLL for each record retrieved. For more information, see [“Using the Data Mining Adapter” on page 357](#).

To import XML data using the Analytics Server XML Gateway

- 1 From the Administration Tool toolbar, select File > Import.
The Select ODBC Data Source dialog box appears.
- 2 Select XML from the Connection Type drop-down list.
The Type In Uniform Resource Locator dialog box appears, with the Connection Type set to XML.
- 3 In the URL field, specify the XML data source URL.
The Analytics Server XML Gateway supports all data sources described in the section [“Locating the XML URL” on page 347](#).
URLs can include repository or session variables. If you click the Browse button, the Select XML File dialog box appears, from which you can select a single file. For more information about variables, see [Chapter 14, “Using Variables in the Analytics Repository.”](#)
- 4 Optionally, type either an Extensible Stylesheet Language Transformations (XSLT) file or XPath expression.
Use the Browse button to browse for XSLT source files.
- 5 Type an optional user ID and password in the appropriate fields for connections to HTTP sites that employ the HTTP Basic Authentication security mode.
In addition to HTTP Basic Authentication security mode, the Analytics Server XML Gateway also supports Secure HTTP protocol and Integrated Windows Authentication (for Windows 2000), formerly called NTLM or Windows NT Challenge/Response authentication.
- 6 Click OK to open the Import dialog box.
- 7 Select the tables and columns and check the type of metadata you want to import.
The default setting imports all objects and all metadata.
- 8 Click Import to begin the import process.
- 9 In the Connection Pool dialog box, type a name and optional description for the connection on the General tab. See [“Setting Up Connection Pools” on page 65](#) for additional details.
- 10 Click the XML tab to set additional connection properties, including the URL refresh interval and the length of time to wait for a URL to load before timing out.
Because XML data sources are typically updated frequently and in real time, the Analytics Server XML Gateway allows users to specify a refresh interval for these data sources.
For more information, see [“About the Refresh Interval for XML Data Sources” on page 276](#).
The default time-out interval for queries (URL loading time-out) is 15 minutes.

- 11 Click OK to complete the import.
- 12 For additional control over the XML data sources, you can specify an XSLT file or an XPath expression for individual tables in the data sources from the Physical Table dialog box. If specified, these entries are used to overwrite corresponding XSLT or XPath entries in the Connection Pool for the respective physical tables.

Analytics Server XML Gateway Example

The following sample XML data document (mytest.xml) references an XML schema contained in an external file. The schema file is shown following the data document. The generated XML schema information available for import to the repository is shown at the end.

```
<?xml version="1.0"?>
<test xmlns="x-schema:mytest_sch.xml">

  <row>
    <p1>0</p1>
    <p2 width="5">
      <p3>hi</p3>
      <p4>
        <p6>xx0</p6>
        <p7>yy0</p7>
      </p4>
      <p5>zz0</p5>
    </p2>
  </row>

  <row>
    <p1>1</p1>
    <p2 width="6">
      <p3>how are you</p3>
      <p4>
        <p6>xx1</p6>
        <p7>yy1</p7>
      </p4>
      <p5>zz1</p5>
    </p2>
  </row>

  <row>
    <p1>a</p1>
    <p2 width="7">
      <p3>hi</p3>
      <p4>
        <p6>xx2</p6>
        <p7>yy2</p7>
      </p4>
      <p5>zz2</p5>
    </p2>
  </row>
```

```
<row>
<p1>b</p1>
<p2 width="8">
  <p3>how are they</p3>
  <p4>
    <p6>xx3</p6>
    <p7>yy3</p7>
  </p4>
  <p5>zz2</p5>
</p2>
</row>
</test>
```

The corresponding schema file follows:

```
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="test" content="elementOnly" order="many">
    <element type="row"/>
  </ElementType>
  <ElementType name="row" content="elementOnly" order="many">
    <element type="p1"/>
    <element type="p2"/>
  </ElementType>
  <ElementType name="p2" content="elementOnly" order="many">
    <AttributeType name="width" dt:type="int" />
    <attribute type="width" />
    <element type="p3"/>
    <element type="p4"/>
    <element type="p5"/>
  </ElementType>
  <ElementType name="p4" content="elementOnly" order="many">
    <element type="p6"/>
    <element type="p7"/>
  </ElementType>
  <ElementType name="p1" content="textOnly" dt:type="string"/>
  <ElementType name="p3" content="textOnly" dt:type="string"/>
  <ElementType name="p5" content="textOnly" dt:type="string"/>
  <ElementType name="p6" content="textOnly" dt:type="string"/>
  <ElementType name="p7" content="textOnly" dt:type="string"/>
</Schema>
```

The name of the table generated from the preceding XML data document (mytest.xml) would be mytest and the column names would be p1, p3, p6, p7, p5, and width.

In addition, to preserve the context in which each column occurs in the document and to distinguish between columns derived from XML elements with identical names but appearing in different contexts, a list of fully qualified column names is generated, based on the XPath proposal of the World Wide Web Consortium, as follows:

```
//test/row/p1
//test/row/p2/p3
//test/row/p2/p4/p6
```

```
//test/row/p2/p4/p7
//test/row/p2/p5
//test/row/p2@width
```

The following example is a more complex example that demonstrates the use of nested table structures in an XML document. Note that you may optionally omit references to an external schema file, in which case all elements would be treated as being of the Varchar character type.

```
===Invoice.xml===
<INVOICE>
  <CUSTOMER>
    <CUST_ID>1</CUST_ID>
    <FIRST_NAME>Nancy</FIRST_NAME>
    <LAST_NAME>Fuller</LAST_NAME>
    <ADDRESS>
      <ADD1>507 - 20th Ave. E., </ADD1>
      <ADD2>Apt. 2A</ADD2>
      <CITY>Seattle</CITY>
      <STATE>WA</STATE>
      <ZIP>98122</ZIP>
    </ADDRESS>
  <PRODUCTS>
    <CATEGORY>
      <CATEGORY_ID>CAT1</CATEGORY_ID>
      <CATEGORY_NAME>NAME1</CATEGORY_NAME>
    <ITEMS>
      <ITEM>
        <ITEM_ID>1</ITEM_ID>
        <NAME></NAME>
        <PRICE>0.50</PRICE>
        <QTY>2000</QTY>
      </ITEM>
      <ITEM>
        <ITEM_ID>2</ITEM_ID>
        <NAME>SPRITE</NAME>
        <PRICE>0.30</PRICE>
        <QTY></QTY>
      </ITEM>
    </ITEMS>
  </CATEGORY>
  <CATEGORY>
    <CATEGORY_ID>CAT2</CATEGORY_ID>
    <CATEGORY_NAME>NAME2</CATEGORY_NAME>
  <ITEMS>
    <ITEM>
      <ITEM_ID>11</ITEM_ID>
      <NAME>ACoke</NAME>
      <PRICE>1.50</PRICE>
      <QTY>3000</QTY>
    </ITEM>
    <ITEM>
      <ITEM_ID>12</ITEM_ID>
      <NAME>SOME SPRITE</NAME>
      <PRICE>3.30</PRICE>
    </ITEM>
  </ITEMS>
</PRODUCTS>
</CUSTOMER>
</INVOICE>
```



```

        <QTY>2000</QTY>
      </I TEM>
    </I TEMS>
  </CATEGORY>
</PRODUCTS>
</CUSTOMER>
<CUSTOMER>
  <CUST_I D>2</CUST_I D>
  <FI RST_NAME>Andrew</FI RST_NAME>
  <LAST_NAME>Carnegi e</LAST_NAME>
  <ADDRESS>
    <ADD1>2955 Campus Dr. </ADD1>
    <ADD2>Ste. 300</ADD2>
    <CI TY>San Mateo</CI TY>
    <STATE>CA</STATE>
    <ZI P>94403</ZI P>
  </ADDRESS>
  <PRODUCTS>
    <CATEGORY>
      <CATEGORY_I D>CAT22</CATEGORY_I D>
      <CATEGORY_NAME>NAMEA1</CATEGORY_NAME>
      <I TEMS>
        <I TEM>
          <I TEM_I D>122</I TEM_I D>
          <NAME>DDDCOKE</NAME>
          <PRI CE>11. 50</PRI CE>
          <QTY>2</QTY>
        </I TEM>
        <I TEM>
          <I TEM_I D>22</I TEM_I D>
          <NAME>PSPRI TE</NAME>
          <PRI CE>9. 30</PRI CE>
          <QTY>1978</QTY>
        </I TEM>
      </I TEMS>
    </CATEGORY>
    <CATEGORY>
      <CATEGORY_I D>CAT24</CATEGORY_I D>
      <CATEGORY_NAME>NAMEA2</CATEGORY_NAME>
      <I TEMS>
        <I TEM>
          <I TEM_I D>19</I TEM_I D>
          <NAME>SOME COKE</NAME>
          <PRI CE>1. 58</PRI CE>
          <QTY>3</QTY>
        </I TEM>
        <I TEM>
          <I TEM_I D>15</I TEM_I D>
          <NAME>DI ET SPRI TE</NAME>
          <PRI CE>9. 30</PRI CE>
          <QTY>12000</QTY>
        </I TEM>
      </I TEMS>
    </CATEGORY>
  </PRODUCTS>
</CUSTOMER>

```

```

    </PRODUCTS>
  </CUSTOMER>
<CUSTOMER>
  <CUST_ID>3</CUST_ID>
  <FIRST_NAME>Margaret</FIRST_NAME>
  <LAST_NAME>Leverling</LAST_NAME>
  <ADDRESS>
    <ADD1>722 Moss Bay Blvd. </ADD1>
    <ADD2> </ADD2>
    <CITY>Kirkland</CITY>
    <STATE>WA</STATE>
    <ZIP>98033</ZIP>
  </ADDRESS>
  <PRODUCTS>
    <CATEGORY>
      <CATEGORY_ID>CAT31</CATEGORY_ID>
      <CATEGORY_NAME>NAMEA3</CATEGORY_NAME>
      <ITEMS>
        <ITEM>
          <ITEM_ID>13</ITEM_ID>
          <NAME>COKE33</NAME>
          <PRICE>30.50</PRICE>
          <QTY>20033</QTY>
        </ITEM>
        <ITEM>
          <ITEM_ID>23</ITEM_ID>
          <NAME>SPRITE33</NAME>
          <PRICE>0.38</PRICE>
          <QTY>20099</QTY>
        </ITEM>
      </ITEMS>
    </CATEGORY>
    <CATEGORY>
      <CATEGORY_ID>CAT288</CATEGORY_ID>
      <CATEGORY_NAME>NAME H</CATEGORY_NAME>
      <ITEMS>
        <ITEM>
          <ITEM_ID>19</ITEM_ID>
          <NAME>COLA</NAME>
          <PRICE>1.0</PRICE>
          <QTY>3</QTY>
        </ITEM>
        <ITEM>
          <ITEM_ID>18</ITEM_ID>
          <NAME>MY SPRITE</NAME>
          <PRICE>8.30</PRICE>
          <QTY>123</QTY>
        </ITEM>
      </ITEMS>
    </CATEGORY>
  </PRODUCTS>
</CUSTOMER>
</INVOICE>

```

The generated XML schema shown next consists of one table (INVOICE) with the following column names and their corresponding fully qualified names.

Column	Fully Qualified Name
ADD1	//I NVOI CE/CUSTOMER/ADDRESS/ADD1
ADD2	//I NVOI CE/CUSTOMER/ADDRESS/ADD2
CITY	//I NVOI CE/CUSTOMER/ADDRESS/CI TY
STATE	//I NVOI CE/CUSTOMER/ADDRESS/STATE
ZIP	//I NVOI CE/CUSTOMER/ADDRESS/ZI P
CUST_ID	//I NVOI CE/CUSTOMER/CUST_ID
FIRST_NAME	//I NVOI CE/CUSTOMER/FI RST_NAME
LAST_NAME	//I NVOI CE/CUSTOMER/LAST_NAME
CATEGORY_ID	//I NVOI CE/CUSTOMER/PRODUCTS/CATEGORY/CATEGORY_ID
CATEGORY_NAME	//I NVOI CE/CUSTOMER/PRODUCTS/CATEGORY/CATEGORY_NAME
ITEM_ID	//I NVOI CE/CUSTOMER/PRODUCTS/CATEGORY/I TEMS/I TEM/I TEM_ID
NAME	//I NVOI CE/CUSTOMER/PRODUCTS/CATEGORY/I TEMS/I TEM/NAME
PRICE	//I NVOI CE/CUSTOMER/PRODUCTS/CATEGORY/I TEMS/I TEM/PRI CE
QTY	//I NVOI CE/CUSTOMER/PRODUCTS/CATEGORY/I TEMS/I TEM/QTY

Only tags with values are extracted as columns. An XML query generates fully qualified tag names, to help make sure that appropriate columns are retrieved.

These are the results of a sample query against the INVOICE table.

```
select first_name, last_name, price, qty, name from invoice
```

```
-----
FIRST_NAME  LAST_NAME      PRI CE  QTY  NAME
-----
Andrew      Carnegi e      1. 58   3    SOME COKE
Andrew      Carnegi e     11. 50   2    DDDCOKE
Andrew      Carnegi e      9. 30  12000 DI ET SPRI TE
Andrew      Carnegi e      9. 30   1978  PSPRI TE
Margar      Leverl ing     0. 38  20099  SPRI TE33
Margar      Leverl ing     1. 0     3     COLA
Margar      Leverl ing    30. 50  20033  COKE33
Margar      Leverl ing     8. 30   123   MY SPRI TE
Nancy       Ful l er       0. 30         SPRI TE
Nancy       Ful l er       0. 50   2000
Nancy       Ful l er       1. 50   3000  ACOKE
Nancy       Ful l er       3. 30   2000  SOME SPRI TE
-----
```

```
Row count: 12
```

Accessing HTML Tables

The Analytics Server XML Gateway also supports the use of tables in HTML files as a data source. The HTML file may be identified as a URL pointing to a file on the internet (including intranet or extranet) or as a file on a local or network drive.

Even though tables, defined by the `<table>` and `</table>` tag pair, are native constructs of the HTML 4.0 specification, they are often used by Web designers as a general formatting device to achieve specific visual effects rather than as a data structure. The Analytics Server XML Gateway is currently the most effective in extracting tables that include specific column headers, defined by `<th>` and `</th>` tag pairs.

For tables that do not contain specific column headers, the Analytics Server XML Gateway employs some simple heuristics to make a best effort to determine the portions of an HTML file that appear to be genuine data tables.

The following is a sample HTML file with one table.

```
<html >
  <body>
    <table border=1 cell padding=2 cell spacing=0>
      <tr>
        <th colspan=1>Transacti on</th>
        <th colspan=2>Measurements</th>
      </tr>
      <tr>
        <th>Qual i ty</th>
        <th>Count</th>
        <th>Percent</th>
      </tr>
      <tr>
        <td>Fai l ed</td>
        <td>66, 672</td>
        <td>4. 1%</td>
      </tr>
      <tr>
        <td>Poor</td>
        <td>126, 304</td>
        <td>7. 7%</td>
      </tr>
      <tr>
        <td>Warni ng</td>
        <td>355, 728</td>
        <td>21. 6%</td>
      </tr>
      <tr>
        <td>OK</td>
        <td>1, 095, 056</td>
        <td>66. 6%</td>
      </tr>
      <tr>
        <td colspan=1>Grand Total </td>
        <td>1, 643, 760</td>
        <td>100. 0%</td>
      </tr>
    </table>
  </body>
</html>
```

```

        </tr>
      </table>
    </body>
  </html>

```

The table name is derived from the HTML filename, and the column names are formed by concatenating the headings (defined by the `<th>` and `</th>` tag pairs) for the corresponding columns, separated by an underscore.

Assuming that our sample file is named *18.htm*, the table name would be *18_0* (because it is the first table in that HTML file), with the following column names and their corresponding fully qualified names.

Column	Fully Qualified Name
Transaction_Quality	\\18_0\Transaction_Quality
Measurements_Count	\\18_0\Measurements_Count
Measurements_Percent	\\18_0\Measurements_Percent

If the table column headings appear in more than one row, the column names are formed by concatenating the corresponding field contents of those header rows.

For tables without any heading tag pairs, the Analytics Server XML Gateway assumes the field values (as delimited by the `<td>` and `</td>` tag pairs) in the first row to be the column names. The columns are named by the order in which they appear (c0, c1, and so on).

For additional examples of XML, see [“XML Examples” on page 362](#).

Using the Data Mining Adapter

The Data Mining Adapter is an extension of the Analytics Server XML Gateway. It allows you to selectively access external data sources by calling an executable file or DLL API for each record retrieved.

The Data Mining Adapter can only be used for a table in a logical join with another table acting as the driving table. The table with the Data Mining Adapter receives parameterized queries from a driving table through some logical joins. The table with the Data Mining Adapter is not a table that physically exists in a back-end database. Instead, the adapter uses the column values in the WHERE clauses of the parameterized queries as its input column parameters, and generates values for those columns (the output columns) not in the WHERE clauses. For information about how to set up the logical joins, see [“Specifying a Driving Table” on page 132](#).

The Data Mining Adapter operates in the following ways:

- **Calls a DLL file.** The Data Mining Adapter allows you to specify a DLL, a shared object, or a shared library that implements the Data Mining Adapter API. At run time, the adapter loads the DLL and calls the API that retrieves records one row at a time. The query results are returned to the XML gateway through an API parameter.

- **Calls an executable file.** The Data Mining Adapter allows you to specify an executable file. At run time, the adapter executes the file and retrieves records from it one row at a time. You also specify the delimiters that demarcate the column values in the output file.

You specify one executable file or DLL for each table.

Using a DLL File to call the Data Mining Adapter API

The API currently consists of only one function. It takes in the values of the input columns in the parameterized queries, plus the meta information of both the input and the output columns. On return, the API places the values of the output columns in the outputColumnValueBuffer. All buffers are allocated by the caller.

Refer to IterativeGatewayDll.h for the definition of the datatype and structures used in this API. You can find this file at the following path:

[*installation root*]\Sample\TestExternalGatewayDll\IterativeGatewayDll.h

Table 50 provides a description of the API elements.

Table 50. API Elements

Element	Description
modelId	An optional argument that you can specify in the Search Utility field in the XML tab of the Physical Table dialog box.
inputColumnCount	The number of input columns.
pInputColumnMetaInfoArray	An array of meta information for the input columns. SiebelAnalyticColumnMetaInfo is declared in the public header file IterativeGatewayDll.h (installed with Siebel Business Analytics).
inputColumnValueBuffer	A buffer of bytes containing the value of the input columns. The actual size of each column value is specified in the columnWidth field of the SiebelAnalyticColumnMetaInfo. The column values are placed in the buffer in the order in which the columns appear in the pInputColumnMetaInfoArray.
OutputColumnCount	The number of output columns.
pOutputColumnMetaInfoArray	An array of meta column information for the output column. SiebelAnalyticColumnMetaInfo is declared in the public header file IterativeGatewayDll.h (installed with Siebel Business Analytics). The caller of the API provides the column name, and the callee sets the data type of the column (currently only VarCharData is supported) and the size of the column value.
outputColumnValueBuffer	A buffer of bytes containing the value of the output columns. The actual size of each column value is specified in the columnWidth field of the SiebelAnalyticColumnMetaInfo. The column values must be placed in the buffer in the order in which the columns appear in the pOutputColumnMetaInfoArray.

Sample Implementation

A sample implementation of the Data Mining Adapter API is provided for all supported platforms in the Sample subdirectory of the Siebel Business Analytics installation folder (\Siebel Analytics). The following files are included in the example:

- hpacc.mak (a HPUX make file for building the sample)
- IterativeGatewayDll.h (a header file to be included in your DLL)
- ReadMe.txt (a text file that describes the Data Mining Adapter API)
- StdAfx.cpp (a Windows-specific file)
- StdAfx.h (a Windows-specific header file)
- sunpro.mak (a Solaris make file for building the sample)
- TestExternalGatewayDll.cpp (the sample implementation of the DLL)
- TestExternalGatewayDll.dsp (a Microsoft Visual C++ project file for building the sample)
- TestLibraryUnix.cpp (a test drive that load up the DLL on the UNIX platforms)
- xIC50.mak (an AIX make file for building the sample)

Using ValueOf() Expressions

You can use ValueOf() expressions in the command line arguments to pass any additional parameters to the executable file or DLL API.

The following example shows how to pass a user ID and password to an executable file:

```
executable_name valueof(USERID) valueof(PASSWORD)
```

Specifying Column Values (Executable File)

When you specify an executable file, you can pass in the column values to the executable file by bracketing the column names with the \$() marker.

For example, suppose there is a table containing the columns Car_Loan, Credit, Demand, Score, and Probability. The values of the input columns Car_Loan, Credit, and Demand come from other tables through join relationships. The values of the output columns Score and Probability are to be returned by the executable file. The command line would look like the following:

```
executable_name $(Car_Loan) $(Credit) $(Demand)
```

Each time the executable file is called, it returns one row of column values. The column values are output in a single-line demarcated by the delimiter that you specify.

By default, the executable is expected to output to the stdout. Alternatively, you can direct the Data Mining Adapter to read the output from a temporary output file passed to the executable as an argument by specifying a placeholder, \$(NO_OUT_TEMP_FILE) to which the executable outputs the result line. When the Data Mining Adapter invokes the executable, the placeholder \$(NO_OUT_TEMP_FILE) is substituted by a temporary filename generated at runtime. This is demonstrated in the following example:

```
executable_name $(Car_Loan) $(Credit) $(Demand) $(NO_OUT_TEMP_FILE)
```

The values of the columns that are not inputs to the executable file will be output first, in the unsorted order in which they appear in the physical table. In the preceding example, the value of the Score column will be followed by the value of the Probability column.

If the executable file outputs more column values than the number of noninput columns, the Data Mining Adapter will attempt to read the column values according to the unsorted column order of the physical table. If these are in conflict with the values of the corresponding input columns, the values returned from the executable file will be used to override the input columns.

The data length of each column in the delimited query output must not exceed the size specified for that column in the physical table.

Configuring the Data Mining Adapter

Use this procedure to configure the Data Mining Adapter.

To configure the Data Mining Adapter

- 1 In the Administration Tool, create a database, select XML Server as the database type, and then click OK.

For information about creating a database, see [“Creating a Database Object Manually in the Physical Layer” on page 62](#).

- 2 Configure the connection pool.

NOTE: Do not type information into any field in the XML tab of the Connection Pool dialog box. The empty fields indicate to the Analytics Server that the Data Mining Adapter functionality will be invoked.

- a Right-click the database you created in Step 1, and then select New Object > Connection Pool.

- b In the General tab, type a name for the connection pool.

The call interface defaults to XML.

- c Type a data source name, and then click OK.

- 3 Right-click the database you created in Step 1, and then select New Object > Table.

- 4 In the Physical Table dialog box, click the XML tab.

- 5 In the XML tab, complete one of the following tasks:

- Select Executable, type the path to the executable file in the Search Utility field, and specify the delimiter for the output values.
- Select DLL and type the path to the DLL in the Search Utility field.

To include spaces in the path, enclose the path in quotation marks. For example:

"C:\Program Files\Siebel Analytics\Bin\SADDataMining.dll"

All characters appearing after the DLL path are passed to the API as a modelid string. You can use the modelid string to pass static or dynamic parameters to the DLL through the API. For example:

"C:\Program Files\Siebel Analytics\Bin\SADDataMining.dll" VALUEOF(Model 1)
VALUEOF(Model 2)

Using XML ODBC

Using the XML ODBC database type, you can access XML data sources through an ODBC interface. The data types of the XML elements representing physical columns in physical tables are derived from the data types of the XML elements as defined in the XML schema. In the absence of a proper XML schema, the default data type of string is used. Data Type settings in the Physical layer will not override those defined in the XML data sources. When accessing XML data without XML schema, use the CAST operator to perform data type conversions in the Business Model and Mapping layer of the Administration Tool.

To import XML data using ODBC

- 1 To access XML data sources through ODBC, you need to license and install an XML ODBC driver.
- 2 Next, create ODBC DSNs that point to the XML data sources you want to access, making sure you select the XML ODBC database type.
- 3 From the File menu, choose Import > from Database.
- 4 Follow the instructions in the dialog boxes to import the ODBC DSNs into the repository.

CAUTION: Make sure you select the Synonyms option in the Import dialog box.

XML ODBC Example

This is an example of an XML ODBC data source in the Microsoft ADO persisted file format. Note that both the data and the schema could be contained inside the same document.

```
<xml xml ns: s=' uui d: BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882'
  xml ns: dt=' uui d: C2F41010-65B3-11d1-A29F-00AA00C14882'
  xml ns: rs=' urn: schemas-microsoft-com: rowset'
  xml ns: z=' #RowsetSchema' >
<s: Schema id=' RowsetSchema' >
  <s: ElementType name=' row' content=' el tOnly' rs: CommandTimeout=' 30'
rs: updatable=' true' >
    <s: AttributeType name=' ShipperID' rs: number=' 1' rs: writeunknown=' true'
rs: basecatalog=' Northwind' rs: basetable=' Shippers'
      rs: basecolumn=' ShipperID' >
      <s: datatype dt: type=' i 2' dt: maxLength=' 2' rs: precision=' 5'
rs: fixedlength=' true' rs: maybenull=' false' />
    </s: AttributeType>
    <s: AttributeType name=' CompanyName' rs: number=' 2' rs: writeunknown=' true'
```

```

rs: basecatalog='Northwind' rs: basetable='Shippers'
  rs: basecolumn='CompanyName' >
    <s: datatype dt: type='string' rs: dbtype='str' dt: maxLength='40'
rs: maybenull='false' />
  </s: AttributeType>
  <s: AttributeType name='Phone' rs: number='3' rs: nullable='true'
rs: writeunknown='true' rs: basecatalog='Northwind'
  rs: basetable='Shippers' rs: basecolumn='Phone' >
    <s: datatype dt: type='string' rs: dbtype='str' dt: maxLength='24'
rs: fixedlength='true' />
  </s: AttributeType>
  <s: extends type='rs: rowbase' />
</s: ElementType>
</s: Schema>

<rs: data>
  <z: row ShipperID='1' CompanyName='Speedy Express' Phone='(503) 555-9831' />
  <z: row ShipperID='2' CompanyName='United Package' Phone='(503) 555-3199' />
  <z: row ShipperID='3' CompanyName='Federal Shipping' Phone='(503) 555-9931' />
</rs: data>
</xml>

```

XML Examples

The following XML documents provide examples of several different situations and explain how the Analytics Server XML access method handles those situations.

- The XML documents *83.xml* and *8_sch.xml* demonstrate the use of the same element declarations in different scope. For example, `<p3>` could appear within `<p2>` as well as within `<p4>`.

Because the element `<p3>` in the preceding examples appears in two different scopes, each element is given a distinct column name by appending an index number to the second occurrence of the element during the Import process. In this case, the second occurrence becomes `p3_1`. If `<p3>` occurs in additional contexts, they become `p3_2`, `p3_3`.

- XML documents *83.xml* and *84.xml* demonstrate that multiple XML files can share the same schema (*8_sch.xml*).
- Internet Explorer version 5 and higher supports HTML documents containing embedded XML fragments called XML islands.

The XML document *island2.htm* demonstrates a simple situation where multiple XML data islands, and therefore multiple tables, could be generated from one document. One table is generated for each instance of an XML island. Tables are distinguished by appending an appropriate index to the document name. For *island2.htm*, the two XML tables generated would be *island2_0* and *island2_1*.

83.xml

```
===83.xml===
```

```

<?xml version="1.0"?>
<test xmlns="x-schema: 8_sch. xml ">|
<row>
<p1>0</p1>
<p2 width="5" height="2">
  <p3>hi </p3>
  <p4>
    <p3>hi </p3>
    <p6>xx0</p6>
    <p7>yy0</p7>
  </p4>
  <p5>zz0</p5>
</p2>
</row>

<row>
<p1>1</p1>
<p2 width="6" height="3">
  <p3>how are you</p3>
  <p4>
    <p3>hi </p3>
    <p6>xx1</p6>
    <p7>yy1</p7>
  </p4>
  <p5>zz1</p5>
</p2>
</row>
</test>

```

8_sch.xml

===8_sch. xml ===

```

<Schema xmlns="urn: schemas-microsoft-com: xml -data" xmlns: dt="urn: schemas-microsoft-com: datatypes">
  <AttributeType name="height" dt: type="int" />
  <ElementType name="test" content="element only" order="many">
    <AttributeType name="height" dt: type="int" />
    <element type="row" />
  </ElementType>
  <ElementType name="row" content="element only" order="many">
    <element type="p1" />
    <element type="p2" />
  </ElementType>
  <ElementType name="p2" content="element only" order="many">
    <AttributeType name="width" dt: type="int" />
    <AttributeType name="height" dt: type="int" />
    <attribute type="width" />
    <attribute type="height" />
    <element type="p3" />
    <element type="p4" />
  </ElementType>

```

```

    <element type="p5"/>
  </ElementType>
  <ElementType name="p4" content="el t0nl y" order="many">
    <element type="p3"/>
    <element type="p6"/>
    <element type="p7"/>
  </ElementType>
  <ElementType name="test0" content="el t0nl y" order="many">
    <element type="row"/>
  </ElementType>
  <ElementType name="p1" content="text0nl y" dt: type="stri ng"/>
  <ElementType name="p3" content="text0nl y" dt: type="stri ng"/>
  <ElementType name="p5" content="text0nl y" dt: type="stri ng"/>
  <ElementType name="p6" content="text0nl y" dt: type="stri ng"/>
  <ElementType name="p7" content="text0nl y" dt: type="stri ng"/>
</Schema>

```

84.xml

===84. xml ===

```

<?xml versi on="1. 0"?>
<test0 xml ns="x-schema: 8_sch. xml ">
  <row>
    <p1>0</p1>
    <p2 wi dth="5" hei ght="2">
      <p3>hi </p3>
      <p4>
        <p3>hi </p3>
        <p6>xx0</p6>
        <p7>yy0</p7>
      </p4>
      <p5>zz0</p5>
    </p2>
  </row>

  <row>
    <p1>1</p1>
    <p2 wi dth="6" hei ght="3">
      <p3>how are you</p3>
      <p4>
        <p3>hi </p3>
        <p6>xx1</p6>
        <p7>yy1</p7>
      </p4>
      <p5>zz1</p5>
    </p2>
  </row>
</test0>

```

Island2.htm

```

===i sl and2. htm===

<HTML>
  <HEAD>
    <TITLE>HTML Document wi th Data Isl and</TITLE>
  </HEAD>
  <BODY>
    <p>This is an example of an XML data island in I.E. 5</p>
    <XML ID="12345">
      <test>
        <row>
          <fi el d1>00</fi el d1>
          <fi el d2>01</fi el d2>
        </row>
        <row>
          <fi el d1>10</fi el d1>
          <fi el d2>11</fi el d2>
        </row>
        <row>
          <fi el d1>20</fi el d1>
          <fi el d2>21</fi el d2>
        </row>
      </test>
    </XML>
    <p>End of fi rst exampl e. </p>
    <XML ID="12346">
      <test>
        <row>
          <fi el d11>00</fi el d11>
          <fi el d12>01</fi el d12>
        </row>
        <row>
          <fi el d11>10</fi el d11>
          <fi el d12>11</fi el d12>
        </row>
        <row>
          <fi el d11>20</fi el d11>
          <fi el d12>21</fi el d12>
        </row>
      </test>
    </XML>
    <p>End of second exampl e. </p>
  </BODY>
</HTML>

```


18 SQL Reference

The Analytics Server accepts SQL SELECT statements from client tools. Additionally, the Administration Tool allows you to define logical tables with complex expressions. This section explains the syntax and semantics for the SELECT statement and for the expressions you can use in the Administration Tool to define logical tables.

SQL Syntax and Semantics

This section explains the syntax and semantics for the SELECT statement. The following topics are included:

- [SELECT Query Specification Syntax on page 367](#)
- [SELECT Usage Notes on page 368](#)
- [SELECT List Syntax on page 368](#)
- [Rules for Queries with Aggregate Functions on page 370](#)

NOTE: The syntax descriptions throughout this guide are not comprehensive. They cover only basic syntax and features unique to the Analytics Server. For a more comprehensive description of SQL syntax, refer to a third-party reference book on SQL or to a reference manual on SQL from your database vendors.

SELECT Query Specification Syntax

The SELECT statement is the basis for querying any structured query language (SQL) database. The Analytics Server accepts logical requests to query objects in a repository, and users (or query tools) make those logical requests with ordinary SQL SELECT statements. The server then translates the logical requests into physical queries against one or more data sources, combines the results to match the logical request, and returns the answer to the end user.

The SELECT statement, or *query specification* as it is sometimes referred to, is the way to query a decision support system through the Analytics Server. A SELECT statement returns a table to the client that matches the query. It is a table in the sense that the results are in the form of rows and columns.

The following is the basic syntax for the SELECT statement. The individual clauses are defined in the subsections that follow.

```
SELECT [DISTINCT] select_list

FROM from_clause

[WHERE search_condition]
```

```
[GROUP BY column {, column}
      [HAVING search_condition]]
[ORDER BY column {, column}]
```

where:

<i>select_list</i>	The list of columns specified in the query. See “SELECT List Syntax” on page 368 .
<i>from_clause</i>	The list of tables in the query, or a catalog folder name. Optionally includes certain join information for the query. See “FROM Clause Syntax” on page 369 .
<i>search_condition</i>	Specifies any combination of conditions to form a conditional test. See “WHERE Clause Syntax” on page 369 .
<i>column</i>	A column (or alias) belonging to a table defined in the data source.

SELECT Usage Notes

The Analytics Server treats the SELECT statement as a logical request. If aggregated data is requested in the SELECT statement, a GROUP BY clause is automatically assumed by the server. Any join conditions supplied in the query are ignored; the join conditions are all predefined in the repository.

The Analytics Server accepts the following SQL syntaxes for comments:

- /* */ C-style comments
- // Double slash for single-line comments
- # Number sign for single-line comments

The Analytics Server supports certain subqueries and UNION, UNION ALL, INTERSECT, and EXCEPT operations in logical requests. This functionality increases the range of business questions that can be answered, eases the formulation of queries, and provides some ability to query across multiple business models.

The Analytics Server supports the following subquery predicates in any conditional expression (for example, within WHERE, HAVING, or CASE statements):

- IN, NOT IN
- >Any, >=Any, =Any, <Any, <=Any, <>Any
- >Some, >=Some, =Some, <Some, <=Some, <>Some
- >All, >=All, =All, <All, <=All, <>All
- EXISTS, NOT EXISTS

SELECT List Syntax

The SELECT list syntax lists the columns in the query.

Syntax:

```
SELECT [DISTINCT] select_list
```

where:

select_list The list of columns specified in the query. All columns need to be from a single business model.

Table names can be included (as *Table.Column*), but are optional unless column names are not unique within a business model.

If column names contain spaces, enclose column names in double quotes. The DISTINCT keyword does not need to be included as the Analytics Server will always do a distinct query.

Columns that are being aggregated do not need to include the aggregation function (such as SUM), as aggregation rules are known to the server and aggregation will be performed automatically.

FROM Clause Syntax

The Analytics Server accepts any valid SQL FROM clause syntax. You can specify the name of a catalog folder instead of a list of tables to simplify FROM clause creation. The Analytics Server determines the proper tables and the proper join specifications based on the columns the query asks for and the configuration of the analytics repository.

WHERE Clause Syntax

The Analytics Server accepts any valid SQL WHERE clause syntax. There is no need to specify any join conditions in the WHERE clause because the joins are all configured within the analytics repository. Any join conditions specified in the WHERE clause are ignored.

The Analytics Server also supports the following subquery predicates in any conditional expression (WHERE, HAVING or CASE statements):

- IN, NOT IN
- >Any, >=Any, =Any, <Any, <=Any, <>Any
- >All, >=All, =All, <All, <=All, <>All
- EXISTS, NOT EXISTS

GROUP BY Clause Syntax

With the Analytics Server's auto aggregation feature, there is no need to submit a GROUP BY clause. When no GROUP BY clause is specified, the GROUP BY specification defaults to all of the nonaggregation columns in the SELECT list. If you explicitly use aggregation functions in the select list, you can specify a GROUP BY clause with different columns and the Analytics Server will compute the results based on the level specified in the GROUP BY clause. For additional details, and some examples of using the GROUP BY clause in queries against the Analytics Server, see ["Rules for Queries with Aggregate Functions" on page 370](#).

ORDER BY Clause Syntax

The Analytics Server accepts any valid SQL ORDER BY clause syntax, including referencing columns by their order in the select list (such as ORDER BY 3, 1, 5).

Rules for Queries with Aggregate Functions

The Analytics Server simplifies the SQL needed to craft aggregate queries. This section outlines the rules that the Analytics Server follows with respect to whether or not a query contains a GROUP BY clause and, if a GROUP BY clause is specified, what results you should expect from the query. The rules outlined in this section apply to all aggregates used in SQL statements (SUM, AVG, MIN, MAX, COUNT(*), and COUNT).

Computing Aggregates of Baseline Columns

A *baseline column* is a column that has no aggregation rule defined in the Aggregation tab of the Logical Column dialog in the repository. Baseline columns map to nonaggregated data at the level of granularity of the logical table to which they belong. If you perform aggregation (SUM, AVG, MIN, MAX, or COUNT) on a baseline column through a SQL request, the Analytics Server calculates the aggregation at the level based on the following rules:

- If there is no GROUP BY clause specified, the level of aggregation is grouped by all of the nonaggregate columns in the SELECT list.
- If there is a GROUP BY clause specified, the level of aggregation is based on the columns specified in the GROUP BY clause.

For example, consider the following query, where the column *revenue* is defined in the repository as a baseline column (no aggregation rules specified in the Logical Column > Aggregation tab):

```
select year, product, sum(revenue)
```

```
from time, products, facts
```

YEAR	PRODUCT	SUM(REVENUE)
1998	Coke	500
1998	Pepsi	600
1999	Coke	600
1999	Pepsi	550
2000	Coke	800
2000	Pepsi	600

This query returns results grouped by *year* and *product*; that is, it returns one row for each product and year combination. The sum calculated for each row is the sum of all the sales for that product in that year. It is logically the same query as the following:

```
select year, product, sum(revenue)
```

```
from time, products, facts
```

```
group by year, product
```

If you change the GROUP BY clause to only group by *year*, then the sum calculated is the sum of all products for the year, as follows:

```
select year, product, sum(revenue)
```

```
from time, products, facts
```

```
group by year
```

YEAR	PRODUCT	SUM(REVENUE)
1998	Coke	1100
1998	Pepsi	1100
1999	Coke	1150
1999	Pepsi	1150
2000	Coke	1400
2000	Pepsi	1400

In this query result set, the sum of *revenue* is the same for each row corresponding to a given year, and that sum represents the total sales for that year. In this case, it is the sales of Coke plus the sales of Pepsi.

If you add a column to the query requesting the COUNT of *revenue*, the Analytics Server calculates the number of records used to calculate the results for each group. In this case, it is a year, as shown in the following example:

```
select year, product, sum(revenue), count(revenue)
from time, products, facts
group by year
```

YEAR	PRODUCT	SUM(REVENUE)	COUNT(REVENUE)
1998	Coke	1100	6000
1998	Pepsi	1100	6000
1999	Coke	1150	6500
1999	Pepsi	1150	6500
2000	Coke	1400	8000
2000	Pepsi	1400	8000

Computing Aggregates of Measure Columns

A *measure column* is a column that has a default aggregation rule defined in the Aggregation tab of the Logical Column dialog in the repository. Measure columns always calculate the aggregation with which they are defined. If you perform explicit aggregation (SUM, AVG, MIN, MAX, or COUNT) on a measure column through a SQL request, you are actually asking for an aggregate of an aggregate. For these *nested* aggregates, the Analytics Server calculates the aggregation based on the following rules:

- A request for a measure column without an aggregate function defined in a SQL statement is always grouped at the level of the nonaggregate columns in the SELECT list, regardless of whether the query specifies a GROUP BY clause.
- If there is no GROUP BY clause specified, the nested aggregate is a grand total of each group determined by all of the nonaggregate columns in the SELECT list.
- If there is a GROUP BY clause specified, the nested aggregation calculates the total for each group as specified in the GROUP BY clause.

For example, consider the following query, where the column SumOfRevenue is defined in the repository as a measure column with a default aggregation rule of SUM (SUM aggregation rule specified in the Aggregation tab of the Logical Column dialog):

```
select year, product, SumOfRevenue, sum(SumOfRevenue)
from time, products, facts
```

YEAR	PRODUCT	SUMofREVENUE	SUM(SUMofREVENUE)
1998	Coke	500	3650
1998	Pepsi	600	3650
1999	Coke	600	3650
1999	Pepsi	550	3650
2000	Coke	800	3650
2000	Pepsi	600	3650

This query returns results grouped by *year* and *product*; that is, it returns one row for each product and year combination. The sum calculated for each row in the SumOfRevenue column is the sum of all the sales for that product in that year because the measure column is always at the level defined by the nonaggregation columns in the query. It is logically the same query as the following:

```
select year, product, SumOfRevenue, sum(SumOfRevenue)
from time, products, facts
group by year, product
```

If you change the GROUP BY clause to only group by *year*, then the sum calculated in the SumOfRevenue column is the sum of each product for the year, and the sum calculated in the SUM(SumOfRevenue) column is total sales of all products for the given year, as follows:

```
select year, product, SumOfRevenue, sum(SumOfRevenue)
from time, products, facts
group by year
```

YEAR	PRODUCT	SUMofREVENUE	SUM(SUMofREVENUE)
1998	Coke	500	1100
1998	Pepsi	600	1100
1999	Coke	600	1150
1999	Pepsi	550	1150
2000	Coke	800	1400
2000	Pepsi	600	1400

In this result set, the sum calculated for each row in the SumOfRevenue column is the sum of all the sales for that product in that year because the measure column is always at the level defined by the nonaggregation columns in the query. The SUM(SumOfRevenue) is the same for each row corresponding to a given year, and that sum represents the total sales for that year. In this case, it is the sales of Coke plus the sales of Pepsi.

Display Function Reset Behavior

A *display function* is a function that operates on the result set of a query. The display functions the Analytics Server supports (RANK, TOP n , BOTTOM n , PERCENTILE, NTILE, MAVG, MEDIAN, and varieties of standard deviation) are specified in the SELECT list of a SQL query. Queries that use display functions conform to the following rules:

- If no GROUP BY clause is specified, the display function operates across the entire result set; that is, the grouping level for the display function is the same as for the query.
- If there is a GROUP BY clause specified, the display function resets its values for each group as specified in the GROUP BY clause.

For example, consider the following query, where SumOfRevenue is defined as a measure column with the default aggregation rule of SUM:

```
select year, product, SumOfRevenue, rank(SumOfRevenue)
from time, products, facts
```

YEAR	PRODUCT	SUMOFREVENUE	RANK(SUMOFREVENUE)
1998	Coke	500	6
1998	Pepsi	600	2
1999	Coke	600	2
1999	Pepsi	550	5
2000	Coke	800	1
2000	Pepsi	600	2

In this query result set, there is no GROUP BY clause specified, so the rank is calculated across the entire result set. The query is logically the same query as the following:

```
select year, product, SumOfRevenue, rank(SumOfRevenue))
from time, products, facts
group by year, product
```

If you change the GROUP BY clause to only group by *year*, then the rank is reset for each year, as follows:

```
select year, product, sum(revenue), rank(sum(revenue))
from time, products, facts
group by year
```

YEAR	PRODUCT	SUM(REVENUE)	RANK(SUM(REVENUE))
1998	Coke	500	2
1998	Pepsi	600	1
1999	Coke	600	1
1999	Pepsi	550	2

2000	Coke	800	1
2000	Pepsi	600	2

In this result set, the rank is reset each time the year changes, and because there are two rows for each year, the value of the rank is always either 1 or 2.

Alternative Syntax

When using an aggregate function, you can calculate a specified level of aggregation using BY within the aggregate function. If you do this, you do not need a GROUP BY clause.

For example, the query:

```
select year, product, revenue, sum(revenue by year) as year_revenue from softdrinks
```

will return the column year_revenue that displays revenue aggregated by year.

The same syntax can be used with display functions. The query:

```
select year, product, revenue, rank(revenue), rank(revenue by year) from softdrinks
order by 1, 5
```

will calculate overall rank of revenue for each product for each year (each row in the entire result set) and also the rank of each product's revenue within each year.

SQL Logical Operators

The following SQL logical operators are used to specify comparisons between expressions.

Between: Used to determine boundaries for a condition. Each boundary is an expression, and the bounds do not include the boundary limits, as in less than and greater than (as opposed to less than or equal to and greater than or equal to). BETWEEN can be preceded with NOT to negate the condition.

In: Specifies a comparison of a column value with a set of values.

Is Null: Specifies a comparison of a column value with the null value.

Like: Specifies a comparison to a literal value. Often used with wildcard characters to indicate any character string match of zero or more characters (%) or a any single character match (_).

Conditional Expressions

The Expressions folder contains building blocks for creating conditional expressions that use CASE, WHEN, THEN and ELSE statements.

CASE (Switch)

This form of the Case statement is also referred to as the CASE (Lookup) form.

Syntax:

```
CASE expression1
  WHEN expression2 THEN expression3
  {WHEN expression. . . THEN expression. . .}
  ELSE expression. . .
END
```

The value of *expression1* is examined, and then the WHEN expressions are examined. If *expression1* matches any WHEN expression, it assigns the value in the corresponding THEN expression.

If none of the WHEN expressions match, it assigns the default value specified in the ELSE expression. If no ELSE expression is specified, the system will automatically add an ELSE NULL.

If *expression1* matches an expression in more than one WHEN clause, only the expression following the first match is assigned.

NOTE: In a CASE statement, AND has precedence over OR.

CASE

Starts the CASE statement. Has to be followed by an expression and one or more WHEN and THEN statements, an optional ELSE statement, and the END keyword.

WHEN

Specifies the condition to be satisfied.

THEN

Specifies the value to assign if the corresponding WHEN expression is satisfied.

ELSE

Specifies the value to assign if none of the WHEN conditions are satisfied. If omitted, ELSE NULL is assumed.

END

Ends the Case statement.

CASE (If)

This form of the CASE statement has the following syntax:

```
CASE
  WHEN search_condition1 THEN expression1
  {WHEN search_condition2 THEN expression2}
  {WHEN search_condition. . . THEN expression. . .}
  ELSE expression
END
```

This evaluates each WHEN condition and if satisfied, assigns the value in the corresponding THEN expression.

If none of the WHEN conditions are satisfied, it assigns the default value specified in the ELSE expression.

If no ELSE expression is specified, the system will automatically add an ELSE NULL.

NOTE: In a CASE statement, AND has precedence over OR.

Unlike the Switch form of the CASE statement, the WHEN statements in the If form allow comparison operators; a WHEN condition of WHEN < 0 THEN 'Under Par' is legal.

CASE

Starts the CASE statement. Has to be followed by one or more WHEN and THEN statements, an optional ELSE statement, and the END keyword.

WHEN

Specifies the condition to be satisfied.

THEN

Specifies the value to assign if the corresponding WHEN expression is satisfied.

ELSE

Specifies the value to assign if none of the WHEN conditions are satisfied. If omitted, ELSE NULL is assumed.

END

Ends the Case statement.

SQL Reference

SQL functions perform various calculations on column values. This section explains the syntax for the functions supported by the Analytics Server. It also explains how to express literals. There are aggregate, string, math, calendar date/time, conversion, and system functions.

The following topics are included:

- [Aggregate Functions on page 377](#)
- [Running Aggregate Functions on page 383](#)
- [String Functions on page 388](#)
- [Math Functions on page 394](#)
- [Calendar Date/Time Functions on page 400](#)
- [Conversion Functions on page 408](#)

- [System Functions on page 410](#)
- [Expressing Literals on page 410](#)

Aggregate Functions

Aggregate functions perform work on multiple values to create summary results. The aggregate functions cannot be used to form nested aggregation in expressions on logical columns that have a default aggregation rule defined in the Aggregation tab of the Logical Column dialog box. To specify nested aggregation, you need to define a column with a default aggregation rule and then request the aggregation of the column in a SQL statement.

Avg

Calculates the average (mean) value of an expression in a result set. Has to take a numeric expression as its argument.

Syntax:

```
AVG (n_expressi on)
```

where:

n_expression Any expression that evaluates to a numerical value.

AvgDistinct

Calculates the average (mean) of all distinct values of an expression. Has to take a numeric expression as its argument.

Syntax:

```
AVG (DI STI NCT n_expressi on)
```

where:

n_expression Any expression that evaluates to a numerical value.

BottomN

Ranks the lowest n values of the expression argument from 1 to n, 1 corresponding to the lowest numerical value. The BOTTOMN function operates on the values returned in the result set.

Syntax:

```
BOTTOMN (n_expressi on, n)
```

where:

n_expression Any expression that evaluates to a numerical value.
n Any positive integer. Represents the bottom number of rankings displayed in the result set, 1 being the lowest rank.

NOTE: A query can contain only one **BOTTOMN** expression.

Count

Calculates the number of rows having a nonnull value for the expression. The expression is typically a column name, in which case the number of rows with nonnull values for that column is returned.

Syntax:

```
COUNT (expression)
```

where:

expression Any expression.

CountDistinct

Adds distinct processing to the COUNT function.

Syntax:

```
COUNT (DISTINCT expression)
```

where:

expression Any expression.

Count (*) (CountStar)

Counts the number of rows.

Syntax:

```
COUNT(*)
```

For example, if a table named Facts contained 200,000,000 rows, the following query would return the following results:

```
SELECT COUNT(*) FROM Facts

COUNT(*)

200000000
```

First

Selects the first returned value of the expression argument. The FIRST function is limited to defining dimension-specific aggregation rules in a repository. You cannot use it in SQL statements.

The FIRST function operates at the most detailed level specified in your explicitly defined dimension. For example, if you have a time dimension defined with hierarchy levels day, month, and year, the FIRST function returns the first day in each level.

You should not use the FIRST function as the first dimension-specific aggregate rule. It might cause queries to bring back large numbers of rows for processing in the Analytics Server causing poor performance.

Syntax:

FIRST (expression)

where:

expression Any expression.

GroupByColumn

For use in setting up aggregate navigation. It specifies the logical columns that define the level of the aggregate data existing in a physical aggregate table.

For example, if an aggregate table contains data grouped by store and by month, specify the following syntax in the content filter (General tab of Logical Source dialog):

```
GROUPBYCOLUMN(STORE, MONTH)
```

The GROUPBYCOLUMN function is only for use in configuring a repository; you cannot use it to form SQL statements.

GroupByLevel

For use in setting up aggregate navigation. It specifies the dimension levels that define the level of the aggregate data existing in a physical aggregate table.

For example, if an aggregate table contains data at the store and month levels, and if you have defined dimensions (Geography and Customers) containing these levels, specify the following syntax in the content filter (General tab of Logical Source dialog):

```
GROUPBYLEVEL (GEOGRAPHY. STORE, CUSTOMERS. MONTH)
```

The GROUPBYLEVEL function is only for use in configuring a repository; you cannot use it to form SQL statements.

Last

Selects the last returned value of the expression. The LAST function is limited to defining dimension-specific aggregation rules in a repository. You cannot use it in SQL statements.

The LAST function operates at the most detailed level specified in your explicitly defined dimension. For example, if you have a time dimension defined with hierarchy levels day, month, and year, the LAST function returns the last day in each level.

You should not use the LAST function as the first dimension-specific aggregate rule. It might cause queries to bring back large numbers of rows for processing in the Analytics Server causing poor performance.

Syntax:

LAST (*expression*)

where:

expression Any expression.

Max

Calculates the maximum value (highest numeric value) of the rows satisfying the numeric expression argument.

Syntax:

MAX (*expression*)

where:

expression Any expression.

The MAX function resets its values for each group in the query, according to the rules outlined in [“Display Function Reset Behavior” on page 373](#).

Median

Calculates the median (middle) value of the rows satisfying the numeric expression argument. When there are an even number of rows, the median is the mean of the two middle rows. This function always returns a double.

Syntax:

MEDI AN (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

The MEDIAN function resets its values for each group in the query, according to the rules outlined in [“Display Function Reset Behavior” on page 373](#).

Min

Calculates the minimum value (lowest numeric value) of the rows satisfying the numeric expression argument.

Syntax:

MIN (*expression*)

where:

expression Any expression.

The MIN function resets its values for each group in the query, according to the rules outlined in [“Display Function Reset Behavior” on page 373](#).

NTile

The NTILE function determines the rank of a value in terms of a user-specified range. It returns integers to represent any range of ranks. In other words, the resulting sorted data set is broken into a number of tiles where there are roughly an equal number of values in each tile.

Syntax:

NTILE (*n_expression*, *n*)

where:

n_expression Any expression that evaluates to a numerical value.

n A positive, nonnull integer that represents the number of tiles.

If the *n_expression* argument is not null, the function returns an integer that represents a rank within the requested range.

NTile with *n*=100 returns what is commonly called the *percentile* (with numbers ranging from 1 to 100, with 100 representing the high end of the sort). This value is different from the results of the Analytics Server percentile function, that conforms to what is called *percent rank* in SQL 92 and returns values from 0 to 1.

Percentile

Calculates a percent rank for each value satisfying the numeric expression argument. The percent rank ranges are from 0 (1st percentile) to 1 (100th percentile), inclusive.

The PERCENTILE function calculates the percentile based on the values in the result set of the query.

Syntax:

PERCENTILE (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

The PERCENTILE function resets its values for each group in the query according to the rules outlined in [“Display Function Reset Behavior” on page 373](#).

Rank

Calculates the rank for each value satisfying the numeric expression argument. The highest number is assigned a rank of 1, and each successive rank is assigned the next consecutive integer (2, 3, 4,...). If certain values are equal, they are assigned the same rank (for example, 1, 1, 1, 4, 5, 5, 7...).

The RANK function calculates the rank based on the values in the result set of the query.

Syntax:

```
RANK (n_expression)
```

where:

n_expression Any expression that evaluates to a numerical value.

The RANK function resets its values for each group in the query according to the rules outlined in [“Display Function Reset Behavior” on page 373](#).

StdDev

The STDDEV function returns the standard deviation for a set of values. The return type is always a double.

Syntax:

```
STDDEV([ALL | DISTINCT] n_expression)
```

where:

n_expression Any expression that evaluates to a numerical value.

- If ALL is specified, the standard deviation is calculated for all data in the set.
- If DISTINCT is specified, all duplicates are ignored in the calculation.
- If nothing is specified (the default), all data is considered.

There are two other functions that are related to STDDEV:

```
STDDEV_POP([ALL | DISTINCT] n_expression)
```

```
STDDEV_SAMP([ALL | DISTINCT] n_expression)
```

STDDEV and STDDEV_SAMP are synonyms.

The STDDEV function resets its values for each group in the query according to the rules outlined in [“Display Function Reset Behavior” on page 373](#).

Sum

Calculates the sum obtained by adding up all values satisfying the numeric expression argument.

Syntax:

```
SUM (n_expression)
```

where:

n_expression Any expression that evaluates to a numerical value.

The SUM function resets its values for each group in the query according to the rules outlined in [“Display Function Reset Behavior” on page 373](#).

SumDistinct

Calculates the sum obtained by adding all of the distinct values satisfying the numeric expression argument.

Syntax:

```
SUM(DI STI NCT n_expressi on)
```

where:

n_expression Any expression that evaluates to a numerical value.

TopN

Ranks the highest *n* values of the expression argument from 1 to *n*, 1 corresponding to the highest numerical value.

The TOPN function operates on the values returned in the result set.

Syntax:

```
TOPN (n_expressi on, n)
```

where:

n_expression Any expression that evaluates to a numerical value.

n Any positive integer. Represents the top number of rankings displayed in the result set, 1 being the highest rank.

A query can contain only one TOPN expression.

The TOPN function resets its values for each group in the query according to the rules outlined in [“Display Function Reset Behavior” on page 373](#).

Running Aggregate Functions

Running aggregate functions are similar to functional aggregates in that they take a set of records as input, but instead of outputting the single aggregate for the entire set of records, they output the aggregate based on records encountered so far.

This section describes the running aggregate functions supported by the Analytics Server.

Mavg

Calculates a moving average (mean) for the last *n* rows of data in the result set, inclusive of the current row.

Syntax:

```
MAVG (n_expression, n)
```

where:

n_expression Any expression that evaluates to a numerical value.

n Any positive integer. Represents the average of the last *n* rows of data.

The MAVG function resets its values for each group in the query, according to the rules outlined in [“Display Function Reset Behavior” on page 373](#).

The average for the first row is equal to the numeric expression for the first row. The average for the second row is calculated by taking the average of the first two rows of data. The average for the third row is calculated by taking the average of the first three rows of data, and so on until you reach the *n*th row, where the average is calculated based on the last *n* rows of data.

MSUM

This function calculates a moving sum for the last *n* rows of data, inclusive of the current row.

The sum for the first row is equal to the numeric expression for the first row. The sum for the second row is calculated by taking the sum of the first two rows of data. The sum for the third row is calculated by taking the sum of the first three rows of data, and so on. When the *n*th row is reached, the sum is calculated based on the last *n* rows of data.

This function resets its values for each group in the query according to the rules described in [“Display Function Reset Behavior” on page 373](#).

Syntax:

```
MSUM (n_expression, n)
```

Where:

n_expression Any expression that evaluates to a numerical value.

n Any positive integer. Represents the sum of the last *n* rows of data.

Example:

The following example shows a query that uses the MSUM function and the query results.

```
select month, revenue, MSUM(revenue, 3) as 3_MO_SUM from sales_subject_area
```

MONTH	REVENUE	3_MO_SUM
JAN	100.00	100.00
FEB	200.00	300.00
MAR	100.00	400.00

APRI L	100. 00	400. 00
MAY	300. 00	500. 00
JUNE	400. 00	800. 00
JULY	500. 00	1200. 00
AUG	500. 00	1400. 00
SEPT	500. 00	1500. 00
OCT	300. 00	1300. 00
NOV	200. 00	1000. 00
DEC	100. 00	600. 00

RSUM

This function calculates a running sum based on records encountered so far. The sum for the first row is equal to the numeric expression for the first row. The sum for the second row is calculated by taking the sum of the first two rows of data. The sum for the third row is calculated by taking the sum of the first three rows of data, and so on.

This function resets its values for each group in the query according to the rules described in [“Display Function Reset Behavior” on page 373](#).

Syntax:

`RSUM (n_expressi on)`

Where:

n_expression Any expression that evaluates to a numerical value.

Example:

The following example shows a query that uses the RSUM function and the query results.

```
select month, revenue, RSUM(revenue) as RUNNI NG_SUM from sales_subj ect_area
```

MONTH	REVENUE	RUNNI NG_SUM
JAN	100. 00	100. 00
FEB	200. 00	300. 00
MAR	100. 00	400. 00
APRI L	100. 00	500. 00
MAY	300. 00	800. 00
JUNE	400. 00	1200. 00
JULY	500. 00	1700. 00
AUG	500. 00	2200. 00
SEPT	500. 00	2700. 00
OCT	300. 00	3000. 00
NOV	200. 00	3200. 00

DEC	100.00	3300.00
-----	--------	---------

RCOUNT

This function takes a set of records as input and counts the number of records encountered so far.

This function resets its values for each group in the query according to the rules described in [“Display Function Reset Behavior” on page 373](#).

Syntax:

`RCOUNT (Expr)`

Where:

Expr An expression of any data type.

Example:

The following example shows a query that uses the RCOUNT function and the query results.

```
select month, profit, RCOUNT(profit) from sales_subject_area where profit > 200.
```

MONTH	PROFIT	RCOUNT (profit)
MAY	300.00	2
JUNE	400.00	3
JULY	500.00	4
AUG	500.00	5
SEPT	500.00	6
OCT	300.00	7

RMAX

This function takes a set of records as input and shows the maximum value based on records encountered so far. The specified data type must be one that can be ordered.

This function resets its values for each group in the query according to the rules described in [“Display Function Reset Behavior” on page 373](#).

Syntax:

`RMAX (expression)`

Where:

expression An expression of any data type. The data type must be one that has an associated sort order.

Example:

The following example shows a query that uses the RMAX function and the query results.

```
select month, profit, RMAX(profit) from sales_subject_area
```

MONTH	PROFIT	RMAX (profit)
JAN	100.00	100.00
FEB	200.00	200.00
MAR	100.00	200.00
APRIL	100.00	200.00
MAY	300.00	300.00
JUNE	400.00	400.00
JULY	500.00	500.00
AUG	500.00	500.00
SEPT	500.00	500.00
OCT	300.00	500.00
NOV	200.00	500.00
DEC	100.00	500.00

RMIN

This function takes a set of records as input and shows the minimum value based on records encountered so far. The specified data type must be one that can be ordered.

This function resets its values for each group in the query according to the rules described in [“Display Function Reset Behavior” on page 373](#).

Syntax:

```
RMIN (expression)
```

Where:

expression An expression of any data type. The data type must be one that has an associated sort order.

Example:

The following example shows a query that uses the RMIN function and the query results.

```
select month, profit, RMIN(profit) from sales_subject_area
```

MONTH	PROFIT	RMIN (profit)
JAN	400.00	400.00
FEB	200.00	200.00
MAR	100.00	100.00
APRIL	100.00	100.00
MAY	300.00	100.00

JUNE	400.00	100.00
JULY	500.00	100.00
AUG	500.00	100.00
SEPT	500.00	100.00
OCT	300.00	100.00
NOV	200.00	100.00
DEC	100.00	100.00

String Functions

String functions perform various character manipulations, and they operate on character strings.

ASCII

Converts a single character string to its corresponding ASCII code, between 0 and 255.

Syntax:

ASCII (*character_expression*)

where:

character_expression Any expression that evaluates to an ASCII character.

If the character expression evaluates to more than one character, the ASCII code corresponding to the first character in the expression is returned.

Bit_Length

Returns the length, in bits, of a specified string. Each Unicode character is 2 bytes in length (equal to 16 bits).

Syntax:

BIT_LENGTH (*character_expression*)

where:

character_expression Any expression that evaluates to character string.

Char

Converts a numerical value between 0 and 255 to the character value corresponding to the ASCII code.

Syntax:

CHAR (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value between 0 and 255.

Char_Length

Returns the length, in number of characters, of a specified string. Leading and trailing blanks are not counted in the length of the string.

Syntax:

```
CHAR_LENGTH (character_expressi on)
```

where:

character_expression Any expression that evaluates to a numerical value between 0 and 255.

Concat

There are two forms of this function. The first form concatenates two character strings. The second form uses the character string concatenation character to concatenate more than two character strings.

Form 1 Syntax:

```
CONCAT (character_expressi on1, character_expressi on2)
```

where:

character_expression Expressions that evaluate to character strings.

Form 2 Syntax:

```
CONCAT (stri ng_expressi on1 || stri ng_expressi on2 || ... stri ng_expressi onxx)
```

where:

string_expression Expressions that evaluate to character strings, separated by the character string concatenation operator || (double vertical bars). The first string is concatenated with the second string to produce an intermediate string, and then this string is concatenated with the next string, and so on.

Insert

Inserts a specified character string into a specified location in another character string.

Syntax:

```
INSERT(character_expressi on, n, m, character_expressi on)
```

where:

<i>character_expression</i>	Any expression that evaluates to a character string.
<i>n</i>	Any positive integer representing the number of characters from the start of the first string where a portion of the second string is inserted.
<i>m</i>	Any positive integer representing the number of characters in the first string to be replaced by the entirety of the second string.

Left

Returns a specified number of characters from the left of a string.

Syntax:

```
LEFT(character_expression, n)
```

where:

<i>character_expression</i>	Any expression that evaluates to a character string.
<i>n</i>	Any positive integer representing the number of characters from the left of the first string that are returned.

Length

Returns the length, in number of characters, of a specified string. The length is returned excluding any trailing blank characters.

Syntax:

```
LENGTH(character_expression)
```

where:

<i>character_expression</i>	Any expression that evaluates to a character string.
-----------------------------	--

Locate

Returns the numerical position of the *character_expression1* in a character expression. If the *character_expression1* is not found in the character expression, the Locate function returns a value of 0. If you want to specify a starting position to begin the search, use the LocateN function instead.

Syntax:

```
LOCATE(character_expression1, character_expression2)
```

where:

<i>character_expression1</i>	Any expression that evaluates to a character string. This is the expression to search for in the character expression.
------------------------------	--

character_expression2 Any expression that evaluates to a character string. This is the expression to be searched.

LocateN

Returns the numerical position of the *character_expression1* in a character expression. This is identical to the *Locate* function, except that the search for the pattern begins at the position specified by an integer argument. If the *character_expression1* is not found in the character expression, the *LocateN* function returns a value of 0. The numerical position to return is determined by counting the first character in the string as occupying position 1, regardless of the value of the integer argument.

Syntax:

```
LOCATE(character_expressi on1, character_expressi on2, n)
```

where:

character_expression1 Any expression that evaluates to a character string. This is the expression to search for in the character expression.

character_expression2 Any expression that evaluates to a character string. This is the expression to be searched.

n Any positive, nonzero integer that represents the starting position to being to look for the locate expression.

Lower

Converts a character string to lower case.

Syntax:

```
LOWER (character_expressi on)
```

where:

character_expression Any expression that evaluates to a character string.

Octet_Length

Returns the bits, in base 8 units (number of bytes), of a specified string.

Syntax:

```
OCTET_LENGTH (character_expressi on)
```

where:

character_expression Any expression that evaluates to a character string.

Position

Returns the numerical position of the `character_expression1` in a character expression. If the `character_expression1` is not found, the function returns 0.

Syntax:

```
POSITION(character_expression1 IN character_expression2)
```

where:

character_expression1 Any expression that evaluates to a character string. Used to search in the second string.

character_expression2 Any expression that evaluates to a character string.

Repeat

Repeats a specified expression *n* times, where *n* is a positive integer.

Syntax:

```
REPEAT(character_expression, n)
```

where:

character_expression Any expression that evaluates to a character string.

n Any positive integer.

Replace

Replaces specified characters from a specified character expression with other specified characters.

Syntax:

```
REPLACE(character_expression, change_expression, replace_with_expression)
```

where:

character_expression Any expression that evaluates to a character string. This first string is the original string.

change_expression Any expression that evaluates to a character string. This second string specifies characters from the first string that will be replaced.

replace_with_expression Any expression that evaluates to a character string. This third string specifies the characters to substitute into the first string.

Right

Returns a specified number of characters from the right of a string.

Syntax:

```
RIGHT(character_expression, n)
```


where:

character_expression Any expression that evaluates to a character string.
n Any positive integer representing the number of characters from the right of the first string that are returned.

Substring

Creates a new string starting from a fixed number of characters into the original string.

Syntax:

```
SUBSTRING (character_expression FROM starting_position)
```

where:

character_expression Any expression that evaluates to a character string.
starting_position Any positive integer representing the number of characters from the start of the string where the result begins.

TrimBoth

Strips specified leading and trailing characters from a character string.

Syntax:

```
TRIM (BOTH 'character' FROM character_expression)
```

where:

character Any single character. If the character part of the specification (and the single quotes) are omitted, a blank character is used as a default.
character_expression Any expression that evaluates to a character string.

TrimLeading

Strips specified leading characters from a character string.

Syntax:

```
TRIM (LEADING 'character' FROM character_expression)
```

where:

character Any single character. If the character part of the specification (and the single quotes) are omitted, a blank character is used as a default.
character_expression Any expression that evaluates to a character string.

TrimTrailing

Strips specified trailing characters from a character string.

Syntax:

```
TRIM (TRAILING 'character' FROM character_expression)
```

where:

<i>character</i>	Any single character. If the character part of the specification (and the single quotes) are omitted, a blank character is used as a default.
<i>character_expression</i>	Any expression that evaluates to a character string.

Upper

Converts a character string to uppercase.

Syntax:

```
UPPER (character_expression)
```

where:

<i>character_expression</i>	Any expression that evaluates to a character string.
-----------------------------	--

Math Functions

The math functions perform mathematical operations.

Abs

Calculates the absolute value of a numerical expression.

Syntax:

```
ABS (n_expression)
```

where:

<i>n_expression</i>	Any expression that evaluates to a numerical value.
---------------------	---

Acos

Calculates the arc cosine of a numerical expression.

Syntax:

```
ACOS (n_expression)
```

where:

n_expression Any expression that evaluates to a numerical value.

Asin

Calculates the arc sine of a numerical expression.

Syntax:

ASIN (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

Atan

Calculates the arc tangent of a numerical expression.

Syntax:

ATAN (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

Atan2

Calculates the arc tangent of y/x, where y is the first numerical expression and x is the second numerical expression.

Syntax:

ATAN2 (*n_expression1*, *n_expression2*)

where:

n_expression (1 and 2) Any expression that evaluates to a numerical value.

Ceiling

Rounds a noninteger numerical expression to the next highest integer. If the numerical expression evaluates to an integer, the Ceiling function returns that integer.

Syntax:

CEILING (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

Cos

Calculates the cosine of a numerical expression.

Syntax:

COS (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

Cot

Calculates the cotangent of a numerical expression.

Syntax:

COT (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

Degrees

Converts an expression from radians to degrees.

Syntax:

DEGREES (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

Exp

Sends the value e to the power specified.

Syntax:

EXP (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

Floor

Rounds a noninteger numerical expression to the next lowest integer. If the numerical expression evaluates to an integer, the FLOOR function returns that integer.

Syntax:

```
FLOOR (n_expression)
```

where:

n_expression Any expression that evaluates to a numerical value.

Log

Calculates the natural logarithm of an expression.

Syntax:

```
LOG (n_expression)
```

where:

n_expression Any expression that evaluates to a numerical value.

Log10

Calculates the base 10 logarithm of an expression.

Syntax:

```
LOG10 (n_expression)
```

where:

n_expression Any expression that evaluates to a numerical value.

Mod

Divides the first numerical expression by the second numerical expression and returns the remainder portion of the quotient.

Syntax:

```
MOD (n_expression1, n_expression2)
```

where:

n_expression (1 and 2) Any expression that evaluates to a numerical value.

Pi

Returns the constant value of pi (the circumference of a circle divided by the diameter of a circle).

Syntax:

PI ()

Power

Takes the first numerical expression and raises it to the power specified in the second numerical expression.

Syntax:

POWER(*n_expression1*, *n_expression2*)

where:

n_expression (1 and 2) Any expression that evaluates to a numerical value.

Radians

Converts an expression from degrees to radians.

Syntax:

RADIANS (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

Rand

Returns a pseudo-random number between 0 and 1.

Syntax:

RAND()

RandFromSeed

Returns a pseudo-random number based on a seed value. For a given seed value, the same set of random numbers are generated.

Syntax:

RAND (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

Round

Rounds a numerical expression to n digits of precision.

Syntax:

```
ROUND (n_expression, n)
```

where:

<i>n_expression</i>	Any expression that evaluates to a numerical value.
<i>n</i>	Any positive integer representing the number of digits of precision with which to round.

Sign

Returns a value of 1 if the numerical expression argument evaluates to a positive number, a value of -1 if the numerical expression argument evaluates to a negative number, and 0 if the numerical expression argument evaluates to zero.

Syntax:

```
SIGN (n_expression)
```

where:

<i>n_expression</i>	Any expression that evaluates to a numerical value.
---------------------	---

Sin

Calculates the sine of a numerical expression.

Syntax:

```
SIN (n_expression)
```

where:

<i>n_expression</i>	Any expression that evaluates to a numerical value.
---------------------	---

Sqrt

Calculates the square root of the numerical expression argument. The numerical expression has to evaluate to a nonnegative number.

Syntax:

```
SQRT (n_expression)
```

where:

<i>n_expression</i>	Any expression that evaluates to a nonnegative numerical value.
---------------------	---

Tan

Calculates the tangent of a numerical expression.

Syntax:

```
TAN (n_expression)
```

where:

n_expression Any expression that evaluates to a numerical value.

Truncate

Truncates a decimal number to return a specified number of places from the decimal point.

Syntax:

```
TRUNCATE (n_expression, n)
```

where:

n_expression Any expression that evaluates to a numerical value.

n Any positive integer representing the number of characters from the right of the decimal place that are returned.

Calendar Date/Time Functions

The calendar date/time functions manipulate data of the data types DATE and DATETIME.

Current_Date

Returns the current date. The date is determined by the system in which the Analytics Server is running.

Syntax:

```
CURRENT_DATE
```

Current_Time

Returns the current time. The time is determined by the system in which the Analytics Server is running.

Syntax:

```
CURRENT_TIME (n)
```


where:

- n* Any integer representing the number of digits of precision with which to display the fractional second. The argument is optional; the function returns the default precision when no argument is specified.

Current_TimeStamp

Returns the current date/timestamp. The timestamp is determined by the system in which the Analytics Server is running.

Syntax:

```
CURRENT_TIMESTAMP (n)
```

where:

- n* Any integer representing the number of digits of precision with which to display the fractional second. The argument is optional; the function returns the default precision when no argument is specified.

Day_Of_Quarter

Returns a number (between 1 and 92) corresponding to the day of the quarter for the specified date.

Syntax:

```
DAY_OF_QUARTER (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

DayName

Returns the day of the week for a specified date.

Syntax:

```
DAYNAME (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

DayOfMonth

Returns the number corresponding to the day of the month for a specified date.

Syntax:

```
DAYOFMONTH (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

DayOfWeek

Returns a number between 1 and 7 corresponding to the day of the week, Sunday through Saturday, for a specified date. For example, the number 1 corresponds to Sunday, and the number 7 corresponds to Saturday.

Syntax:

DAYOFWEEK (*date_expression*)

where:

date_expression Any expression that evaluates to a date.

DayOfYear

Returns the number (between 1 and 366) corresponding to the day of the year for a specified date.

Syntax:

DAYOFYEAR (*date_expression*)

where:

date_expression Any expression that evaluates to a date.

Hour

Returns a number (between 0 and 23) corresponding to the hour for a specified time. For example, 0 corresponds to 12 a.m. and 23 corresponds to 11 p.m.

Syntax:

HOURL (*time_expression*)

where:

time_expression Any expression that evaluates to a time.

Minute

Returns a number (between 0 and 59) corresponding to the minute for a specified time.

Syntax:

MINUTE (*time_expression*)

where:

time_expression Any expression that evaluates to a time.

Month

Returns the number (between 1 and 12) corresponding to the month for a specified date.

Syntax:

```
MONTH (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

Month_Of_Quarter

Returns the number (between 1 and 3) corresponding to the month in the quarter for a specified date.

Syntax:

```
MONTH_OF_QUARTER (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

MonthName

Returns the name of the month for a specified date.

Syntax:

```
MONTHNAME (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

Now

Returns the current timestamp. The NOW function is equivalent to the CURRENT_TIMESTAMP function.

Syntax:

```
NOW ()
```

Quarter_Of_Year

Returns the number (between 1 and 4) corresponding to the quarter of the year for a specified date.

Syntax:

```
QUARTER_OF_YEAR (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

Second

Returns the number (between 0 and 59) corresponding to the seconds for a specified time.

Syntax:

```
SECOND (time_expression)
```

where:

time_expression Any expression that evaluates to a time.

TimestampAdd

The TimestampAdd function adds a specified number of intervals to a specified timestamp. A single timestamp is returned.

Syntax:

```
TimestampAdd (interval, integer-expression, timestamp-expression)
```

where:

interval The specified interval. Valid values are:

SQL_TSI_SECOND

SQL_TSI_MINUTE

SQL_TSI_HOUR

SQL_TSI_DAY

SQL_TSI_WEEK

SQL_TSI_MONTH

SQL_TSI_QUARTER

SQL_TSI_YEAR

integer_expression Any expression that evaluates to an integer.

timestamp_expression The timestamp used as the base in the calculation.

A null integer-expression or a null timestamp-expression passed to this function will result in a null return value.

In the simplest scenario, this function merely adds the specified integer value (integer-expression) to the appropriate component of the timestamp, based on the interval. Adding a week translates to adding seven days, and adding a quarter translates to adding three months. A negative integer value results in a subtraction (going back in time).

An overflow of the specified component (such as more than 60 seconds, 24 hours, twelve months, and so on) necessitates adding an appropriate amount to the next component. For example, when adding to the day component of a timestamp, this function considers overflow and takes into account the number of days in a particular month (including leap years when February has 29 days).

When adding to the month component of a timestamp, this function verifies that the resulting timestamp has a sufficient number of days for the day component. For example, adding 1 month to 2000-05-31 does not result in 2000-06-31 because June does not have 31 days. This function reduces the day component to the last day of the month, 2000-06-30 in this example.

A similar issue arises when adding to the year component of a timestamp having a month component of February and a day component of 29 (that is, last day of February in a leap year). If the resulting timestamp does not fall on a leap year, the function reduces the day component to 28.

These actions conform to the behavior of Microsoft's SQL Server and Oracle's native OCI interface.

The following queries are examples of the TimestampAdd function and its results:

The following query asks for the resulting timestamp when 3 days are added to 2000-02-27 14:30:00. Since February, 2000 is a leap year, the query returns a single timestamp of 2000-03-01 14:30:00.

```

Select TimestampAdd(SQL_TSI_DAY, 3,
Timestamp' 2000-02-27 14:30:00' )
From Employee where employeeid = 2;

```

The following query asks for the resulting timestamp when 7 months are added to 1999-07-31 0:0:0. The query returns a single timestamp of 2000-02-29 00:00:00. Notice the reduction of day component to 29 because of the shorter month of February.

```

Select TimestampAdd(SQL_TSI_MONTH, 7,
Timestamp' 1999-07-31 00:00:00' )
From Employee where employeeid = 2;

```

The following query asks for the resulting timestamp when 25 minutes are added to 2000-07-31 23:35:00. The query returns a single timestamp of 2000-08-01 00:00:00. Notice the propagation of overflow through the month component.

```

Select TimestampAdd(SQL_TSI_MINUTE, 25,
Timestamp' 2000-07-31 23:35:00' )

```

From Employee where employeeid = 2;

CAUTION: The `TIMESTAMPADD` function is turned on by default for Microsoft SQL Server, ODBC, IBM DB2, and Oracle databases. Because DB2 and Oracle semantics do not fully support this function, the answers from this function might not match exactly with what the Analytics Server computes.

TimeStampDiff

The `TimeStampDiff` function returns the total number of specified intervals between two timestamps.

Syntax:

`TimeStampDiff (interval, timestamp-expression1, timestamp-expression2)`

where:

interval The specified interval. Valid values are:

`SQL_TSI_SECOND`

`SQL_TSI_MINUTE`

`SQL_TSI_HOUR`

`SQL_TSI_DAY`

`SQL_TSI_WEEK`

`SQL_TSI_MONTH`

`SQL_TSI_QUARTER`

`SQL_TSI_YEAR`

timestamp_expression1 The timestamp used in the function.

timestamp_expression2 The first timestamp used in the function.

A null timestamp-expression parameter passed to this function will result in a null return value.

This function first determines the timestamp component that corresponds to the specified interval parameter. For example, `SQL_TSI_DAY` corresponds to the day component and `SQL_TSI_MONTH` corresponds to the month component.

The function then looks at the higher order components of both timestamps to calculate the total number of intervals for each timestamp. For example, if the specified interval corresponds to the month component, the function calculates the total number of months for each timestamp by adding the month component and twelve times the year component.

Finally, the function subtracts the first timestamp's total number of intervals from the second timestamp's total number of intervals.

The `TimeStampDiff` function rounds up to the next integer whenever fractional intervals represent a crossing of an interval boundary. For example, the difference in years between 1999-12-31 and 2000-01-01 is one year because the fractional year represents a crossing from one year to the next (that is, 1999 to 2000). By contrast, the difference between 1999-01-01 and 1999-12-31 is zero years because the fractional interval falls entirely within a particular year (that is, 1999).

Microsoft's SQL Server exhibits the same rounding behavior. IBM DB2 always rounds down. Oracle does not implement a generalized timestamp difference function.

When calculating the difference in weeks, the function calculates the difference in days and divides by seven before rounding. Additionally, the function takes into account how the Analytics Server administrator has configured the start of a new week in the NQSSConfig.INI file using the parameter `FIRST_DAY_OF_THE_WEEK` (defaults to Sunday).

For example, with Sunday as the start of the week, the difference in weeks between 2000-07-06 (a Thursday) and 2000-07-10 (the following Monday) results in a value of one week. With Tuesday as the start of the week, however, the function would return zero weeks since the fractional interval falls entirely within a particular week.

When calculating the difference in quarters, the function calculates the difference in months and divides by three before rounding.

IBM DB2 provides a generalized timestamp difference function (`TIMESTAMPDIFF`) but it simplifies the calculation by always assuming a 365-day year, 52-week year, and 30-day month.

TimestampDiff Function and Results Example

The following query asks for a difference in days between timestamps 1998-07-31 23:35:00 and 2000-04-01 14:24:00. It returns a value of 610. Notice that the leap year in 2000 results in an additional day.

```
Select TimestampDiff(SQL_TSI_DAY, TIMESTAMP' 1998-07-31 23:35:00', TIMESTAMP' 2000-04-01 14:24:00') From Employee where employeeid = 2;
```

CAUTION: The `TIMESTAMPDIFF` function is turned on by default for Microsoft SQL Server, ODBC, IBM DB2, and Oracle databases. Because DB2 and Oracle semantics do not fully support this function, the answers from this function might not match exactly with what the Analytics Server computes.

Week_Of_Quarter

Returns a number (between 1 and 13) corresponding to the week of the quarter for the specified date.

Syntax:

```
WEEK_OF_QUARTER (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

Week_Of_Year

Returns a number (between 1 and 53) corresponding to the week of the year for the specified date.

Syntax:

```
WEEK_OF_YEAR (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

Year

Returns the year for the specified date.

Syntax:

```
YEAR (date_expression)
```

where:

date_expression Any expression that evaluates to a date.

Conversion Functions

The conversion functions convert a value from one form to another.

Cast

Changes the data type of an expression or a null literal to another data type. For example, you can cast a customer_name (a data type of Char or Varchar) or birthdate (a datetime literal). The following are the supported data types to which the value can be changed:

```
CHARACTER, VARCHAR, INTEGER, FLOAT, SMALLINT, DOUBLE PRECISION, DATE, TIME,
TIMESTAMP, BIT, BIT VARYING
```

NOTE: Depending on the source data type, some destination types are not supported. For example, if the source data type is a BIT string, the destination data type has to be a character string or another BIT string.

The following describes unique characteristics of the CHAR and VARCHAR data types:

- **Casting to a CHAR data type.** You must use a size parameter. If you do not add a size parameter, a default of 30 will be added. Syntax options appear in the following list:

- It is recommended that you use the following syntax:

```
CAST (expression|NULL AS CHAR(n) )
```

For example, CAST (companyname AS CHAR(35))

- You can use the following syntax:

```
CAST (expression|NULL AS datatype )
```

For example, CAST (companyname AS CHAR)

NOTE: If you use this syntax, Siebel Business Analytics will explicitly convert and store as CAST (expression|NULL AS CHAR(30))

- **Casting to a VARCHAR data type.** The Administration Tool requires that you use a size parameter. If you omit the size parameter, you cannot save the change.

Choose

Takes an arbitrary number of parameters and returns the first item in the list that the user has permission to see.

Syntax:

```
CHOOSE (expression1, expression2, . . . , expressionN)
```

For example, a single query can be written to return security-based revenue numbers for the entire organization. The function could look like the following:

```
choose(L1-Revenue, L2-Revenue, L3-Revenue, L4-Revenue)
```

If the user issuing this function has access to the column L1-Revenue, then that column value would be returned. If the user does not have visibility to the column L1-Revenue but does have visibility to L2-Revenue, then L2-Revenue is returned.

IfNull

Tests if an expression evaluates to a null value, and if it does, assigns the specified value to the expression.

Syntax:

```
IFNULL (expression, value)
```

VALUEOF()

Use the VALUEOF function in an expression builder or filter to reference the value of a repository variable defined in Oracle's Siebel Analytics Administration Tool. You can also use the VALUEOF function when you edit the SQL for a request from the Advanced tab in Oracle's Siebel Business Analytics Answers.

Variables should be used as arguments of the VALUEOF function. Refer to static repository variables by name. For example, to use the value of a static repository variables named prime_begin and prime_end:

```
CASE WHEN "Hour" >= VALUEOF("prime_begin") AND "Hour" < VALUEOF("prime_end") THEN  
  'Prime Time' WHEN ... ELSE ... END
```

You need to refer to a dynamic repository variable by its fully qualified name. If you are using a dynamic repository variable, the names of the initialization block and the repository variable need to be enclosed in double quotes ("), separated by a period, and contained within parentheses. For example, to use the value of a dynamic repository variable named REGION contained in an initialization block named Region Security, this is an example of the proper syntax to use:

```
SalesSubjectArea.Customer.Region =  
VALUEOF("Region Security"."REGION")
```

The names of session variables need to be preceded by NQ_SESSION, separated by a period, and contained within parentheses. If the variable name contains a space, enclose the name in double quotes ("). For example, to use the value of a session variable named REGION, this is an example of the proper syntax to use in an expression builder (filter):

```
"SalesSubjectArea"."Customer"."Region" = VALUEOF(NQ_SESSION.REGION)
```

NOTE: Although using initialization block names with session variables (just as with other repository variables) may work, you should use NQ_SESSION. NQ_SESSION acts like a wild card that matches all initialization block names. This allows the administrator to change the structure of the initialization blocks in a localized manner without impacting reports.

System Functions

The system functions return values relating to the session.

User

Returns the user ID for the analytics repository to which you are logged in.

Syntax:

```
USER ()
```

Database

Returns the name of the Siebel Business Analytics presentation catalog to which you are logged in.

Syntax:

```
DATABASE ()
```

Expressing Literals

A literal is a nonnull value corresponding to a given data type. Literals are typically constant values; that is, they are values that are taken literally *as is*, without changing them at all. A literal value has to comply with the data type it represents.

SQL provides mechanisms for expressing literals in SQL statements. This section describes how to express each type of literal in SQL.

Character Literals

A character literal represents a value of CHARACTER or VARCHAR data type. To express a character literal, surround the character string with single quotes ('). The length of the literal is determined by the number of characters between the single quotes.

Datetime Literals

The SQL 92 standard defines three kinds of typed datetime literals, in the following formats:

DATE 'yyyy-mm-dd'

TIME 'hh:mm:ss'

TIMESTAMP 'yyyy-mm-dd hh:mm:ss'

These formats are fixed and are not affected by the format specified in the NQSConfig.INI file for the parameters DATE_DISPLAY_FORMAT, TIME_DISPLAY_FORMAT, or DATE_TIME_DISPLAY_FORMAT. To express a typed datetime literal, use the keywords DATE, TIME, or TIMESTAMP followed by a datetime string enclosed in single quote marks. Two digits are required for all nonyear components even if the value is a single digit.

Numeric Literals

A numeric literal represents a value of a numeric data type (for example, INTEGER, DECIMAL, and FLOAT). To express a numeric literal, type the number as part of a SQL statement.

Do not surround numeric literals with single quotes; doing so expresses the literal as a character literal.

Integers

To express an integer constant as a literal, type the integer as part of a SQL statement (for example, in the SELECT list). The integer can be preceded with either a plus sign (+) or minus sign (-) to indicate the integer is a positive or negative number, respectively.

Decimal

To express a decimal literal, type a decimal number. The decimal can be preceded with either a plus sign (+) or minus sign (-) to indicate the integer is a positive or negative number, respectively.

Floating Point

To express floating point numbers as literal constants, type a decimal literal followed by the letter 'E' (either uppercase or lowercase) and followed by the plus sign (+) or the minus sign (-) to indicate a positive or negative exponent. No spaces are allowed between the integer, the letter 'E', and the sign of the exponent.

A

Usage Tracking Data Descriptions and Using the Log File Method

The Analytics Server supports the collection of usage tracking data. When usage tracking is enabled, the Analytics Server collects usage tracking data for each query and writes statistics to a usage tracking log file or inserts them directly to a database table.

NOTE: It is recommended that you use Direct Insert instead of writing to a log file.

For more information, see [“Administering Usage Tracking” on page 246](#).

Create Table Scripts for Usage Tracking Data

The SiebelAnalytics\Schema folder contains the following Create Table scripts for DB2, SQL Server and Oracle:

- SAACCT.Oracle.sql for Oracle
- SAACCT.DB2.sql for DB2
- SAACCT.MSSQL.sql for SQL Server

The sample scripts set the usage tracking table name to S_NQ_ACCT. For sites that have Siebel Business Analytics applications, this is the name used in the analytics repository. Sites that build their own repositories should name the usage tracking table to match the name in the corresponding repository.

Loading Usage Tracking Tables with Log Files

It is strongly recommended that you load the Usage Tracking table using the Direct-Insert option. For those customers who have to load the Usage Tracking table from log files, Siebel Business Analytics provides the following sample JavaScript files located in the SiebelAnalytics\Scripts\Common subdirectory:

- sbIAcctLoaderMSSQL for SQL Server
- sbIAcctLoaderOCL.js for Oracle
- sbIAcctLoaderADO.js for other database servers like DB2

These JavaScript files need to be modified for your environment. Comments at the beginning of these files should be used as a guide.

Before extracting usage tracking log file data, you need to create a table to store the data. For more information, see [“Create Table Scripts for Usage Tracking Data” on page 413](#).

NOTE: UNIX installations cannot use these JavaScript files because UNIX operating systems typically do not support some of the advanced functionality used in these scripts. For UNIX installations, please ask your DBA to write scripts that will load your usage tracking log files.

Description of the Usage Tracking Data

Table 51 describes each column in the usage tracking table.

Table 51. Usage Tracking Data

Column	Description
USER_NAME	The name of the user who submitted the query.
REPOSITORY_NAME	The name of the repository the query accesses.
SUBJECT_AREA_NAME	The name of the business model being accessed.
NODE_ID	Reserved for future use.
START_TS	The date and time the logical query was submitted.
START_DT	The date the logical query was submitted.
START_HOUR_MIN	The hour and minute the logical query was submitted.
END_TS	The date and time the logical query finished. The start and end timestamps also reflect any time the query spent waiting for resources to become available.
END_DT	The date the logical query was completed.
END_HOUR_MIN	The hour and minute the logical query was completed.
QUERY_TEXT	The SQL submitted for the query.
SUCCESS_FLG	The completion status of the query: 0 - The query completed successfully with no errors. 1 - The query timed out. 2 - The query failed because row limits were exceeded. 3 - The query failed due to some other reason.
ROW_COUNT	The number of rows returned to the query client.
COMPILE_TIME_SEC	The time in seconds required to compile the query.
TOTAL_TIME_SEC	The time in seconds that the Analytics Server spent working on the query while the client waited for responses to its query requests.
NUM_DB_QUERY	The number of queries submitted to back-end databases in order to satisfy the logical query request. For successful queries (SuccessFlag = 0) this number will be 1 or greater.
CUM_DB_TIME_SEC	The total amount of time in seconds that the Analytics Server waited for back-end physical databases on behalf of a logical query.

Table 51. Usage Tracking Data

Column	Description
CUM_NUM_DB_ROW	The total number of rows returned by the back-end databases.
CACHE_IND_FLG	Default is N. Y indicates a cache hit for the query, N indicates a cache miss.
QUERY_SRC_CD	The source of the request, for example, Drill or Report.
SAW_SRC_PATH	The path name in the Siebel Business Analytics Web Catalog for the request.
SAW_DASHBOARD	The name of the dashboard. Null if the query was not submitted through Oracle's Siebel Intelligence Dashboard.

B

Pharma Mobile Analytics Administration Reference

Pharma Mobile Analytics is a preconfigured Mobile Analytics application for Pharma Sales. Pharma components for Mobile Analytics are installed during Oracle's Siebel Business Analytics installation in the \SiebelAnalyticsData\Disconnected\Pharma directory.

The following is a list of components that have been tailored for Pharma Mobile Analytics and their locations:

- **PharmaDisconnect.rpd.** The Pharma Mobile Analytics repository is tailored for Pharma Mobile Analytics and downloaded to the laptop during synchronization. It is located in the \SiebelAnalyticsData\Disconnected\Pharma\Application directory.
- **PharmaDisconnect.webcat.** Web Catalog tailored for Pharma Mobile Analytics. It is located in the \SiebelAnalyticsData\Disconnected\Pharma\Application directory.
- **Pharma.XML.** Application configuration file tailored for Pharma Mobile Analytics. It is located in the \SiebelAnalyticsData\Disconnected\Pharma directory.

This appendix contains the reference material about the following topics for Pharma Mobile Analytics administrators and system administrators.

- [Sourcing Reports for Pharma Mobile Analytics](#)
- [Data Sets for Pharma Mobile Analytics on page 419](#)

Sourcing Reports for Pharma Mobile Analytics

[Table 52 on page 417](#) contains a list of the sourcing reports (queries) for Pharma Mobile Analytics. These reports issue logical SQL to the Analytics Server to extract the data used to populate the local mobile client schema.

Table 52. Sourcing Reports for Pharma Mobile Analytics

Subject Area	Sourcing Report Name	Target Table
Pharma Promotional Effectiveness	Day Dimension	W_DAY_D
Pharma Promotional Effectiveness	Product Ranking Dimension	W_CON_RANK_D
Pharma Promotional Effectiveness	Brick Dimension	W_AREA_D
Pharma Promotional Effectiveness	Geography Dimension	W_GEO_D

Table 52. Sourcing Reports for Pharma Mobile Analytics

Subject Area	Sourcing Report Name	Target Table
Pharma Promotional Effectiveness	Payor Plan Dimension	W_INS_PLAN_D
Pharma Promotional Effectiveness	Product Hierarchy	W_PRODUCT_DH
Pharma Promotional Effectiveness	Position Hierarchy	W_POSITION_DH
Pharma Promotional Effectiveness	Call Priority List of Value Dimension	W_LOV_D
Pharma Promotional Effectiveness	Product Indication List of Value Dimension	W_LOV_D
Pharma Sales Effectiveness	Period Type List of Value Dimension	W_LOV_D
Pharma Promotional Effectiveness	Person Dimension	W_PERSON_D
Pharma Objective Achievement	Plan Promotion Dimension	W_PLAN_PROMO_D
Pharma Customer Demographics	Contact Primary Address Dimension	W_POSTN_CON_D
Pharma Promotional Effectiveness	Contact Call Activity Fact	W_CON_CALL_F
Pharma Promotional Effectiveness	Contact Call Activity Aggregate	W_CON_CALL_N_A
Pharma Objective Achievement	Contact Objective Fact	W_CON_OBJ_F
Pharma Sales Effectiveness	Indirect Sales Market Level Fact	W_MARKET_IDS_F
Pharma Sales Effectiveness	Indirect Sales Market Level Aggregate	W_MARKET_IDS_N_A
Pharma Sales Effectiveness	Prescription Market Level Fact	W_MARKET_RX_F
Pharma Sales Effectiveness	Prescription Market Level Aggregate	W_MARKET_RX_N_A
Pharma Sales Effectiveness	Direct Sales Fact	W_SYND_DS_F
Pharma Sales Effectiveness	Direct Sales Aggregate	W_SYND_DS_N_A
Pharma Sales Effectiveness	Indirect Sales Fact	W_SYND_IDS_F
Pharma Sales Effectiveness	Indirect Sales Aggregate	W_SYND_IDS_N_A
Pharma Sales Effectiveness	Prescription Sales Fact	W_SYND_RX_F
Pharma Sales Effectiveness	Prescription Sales Aggregate	W_SYND_RX_N_A

Data Sets for Pharma Mobile Analytics

Table 53 on page 419 contains data set information for Oracle's Pharma Mobile Analytics. For more information, see the Pharma configuration file (Pharma.XML).

Table 53. Data Sets for Pharma Mobile Analytics

Data Set Name	Sourcing Report Name	Target Table
Day	Day Dimension	W_DAY_D
ProductRanking	Product Ranking Dimension	W_CON_RANK_D
Geography	Brick Dimension	W_AREA_D
Geography	Geography Dimension	W_GEO_D
PayorProductPosition	Payor Plan Dimension	W_INS_PLAN_D
PayorProductPosition	Product Hierarchy	W_PRODUCT_DH
PayorProductPosition	Position Hierarchy	W_POSITION_DH
CallPriorityLov	Call Priority List of Value Dimension	W_LOV_D
ProductIndicationLov	Product Indication List of Value Dimension	W_LOV_D
PeriodTypeLov	Period Type List of Value Dimension	W_LOV_D
Person	Person Dimension	W_PERSON_D
SalesObjective	Plan Promotion Dimension	W_PLAN_PROMO_D
ContactPrimaryAddress	Contact Primary Address Dimension	W_POSTN_CON_D
ContactCallActivity	Contact Call Activity Fact	W_CON_CALL_F
ContactCallActivity	Contact Call Activity Aggregate	W_CON_CALL_N_A
ContactObjective	Contact Objective Fact	W_CON_OBJ_F
SyndicateData	Indirect Sales Market Level Fact	W_MARKET_IDS_F
SyndicateData	Indirect Sales Market Level Aggregate	W_MARKET_IDS_N_A
SyndicateData	Prescription Market Level Fact	W_MARKET_RX_F
SyndicateData	Prescription Market Level Aggregate	W_MARKET_RX_N_A
SyndicateData	Direct Sales Fact	W_SYND_DS_F
SyndicateData	Direct Sales Aggregate	W_SYND_DS_N_A
SyndicateData	Indirect Sales Fact	W_SYND_IDS_F
SyndicateData	Indirect Sales Aggregate	W_SYND_IDS_N_A
SyndicateData	Prescription Sales Fact	W_SYND_RX_F
SyndicateData	Prescription Sales Aggregate	W_SYND_RX_N_A

C

Mobile Analytics Configuration File Reference

This section lists the mobile application configuration file XML tags with their attributes. The list is in alphabetical order by XML tag and then by attribute under each XML tag.

CAUTION: Tags and attributes are case sensitive. All must appear in lower case.

■ **XML Tag:** <application>

This tag identifies the mobile application.

Parent tag: <remotecfg>

Attributes:

name

The internal name of the mobile application. It is the same name as the mobile application directory described in [“Mobile Application Directory” on page 181](#). It is the same name as the base name of the mobile application configuration file. For example, a mobile application called MyApp would have a configuration file called MyApp.xml that resides in the MyApp directory.

The name must consist of only digits and alphabetic characters. It can be as long as the maximum directory name length allowed by the operating system of the enterprise server.

Type: String

Required: Yes

dir

The name of the mobile application metadata directory described in [“Mobile Application Metadata Directory” on page 181](#).

Type: String

Required: No. Default is the mobile application directory described in [“Mobile Application Directory” on page 181](#).

■ **XML Tag:** <data>

Parent tag: <remotecfg>

Attributes:

dir

The name of the mobile application data directory described in [“Mobile Application Data Directory” on page 181](#).

Type: String

Required: No. Default is the mobile application directory described in [“Mobile Application Directory” on page 181](#).

catalogfolder

The full path of the folder within the enterprise Web Catalog that contains the various sourcing reports. Syntactically, anything that Siebel Business Analytics Web accepts will be valid as long as such a folder actually exists in the Web Catalog and all mobile application end users have read-access to it.

Type: String

Required: Yes

■ XML Tag: <dataset>

Parent tag: <data>

Attributes:

name

An internal name that uniquely identifies the data set. Dependent data sets in the same data set family specify this name in the <dependson> tag.

Type: String

Required: Yes

rank

This attribute indicates the order in which the data set will appear in the advanced area of the Disconnected Analytics Application Manager and the order in which the Disconnected Analytics Application Manager will process the data set. The attribute must be a non-negative integer. Multiple data sets may have the same rank, in which case the Disconnected Analytics Application Manager will display and process them in the ascending alphabetical order of their respective names.

Type: Number

Required: Yes

validitytype

This attribute specifies which validity scheme to use for the data set. The Disconnected Analytics Application Manager uses one of two validation schemes for determining whether to download a given data set: date-based and rolling-period. For the date-based scheme, the Disconnected Analytics Application Manager (potentially) needs to verify that the data set's most recent successful synchronization date exceeds or equals a specified start date (see the start attribute) and that the current date (always) precedes an end date if specified (see the expiry attribute). For the rolling-period scheme, the Disconnected Analytics Application Manager must verify that the data set's most recent successful synchronization date falls within a specified time interval ending at the current date. The attributes period and periodunit define this interval.

Type: Keyword (valid values are date-based and rolling-period)

Required: Yes

start

For the date-based validity option, the data set's most recent synchronization date must equal or exceed this date for the data set to be valid. This only applies to incremental data sets, however; for full data sets, the Disconnected Analytics Application Manager ignores this attribute entirely. Specifying this attribute for an incremental data set necessarily implies that the incremental data set must be downloaded after the full data set in a separate download. To allow the initial downloading of the full data set and all its dependent incremental data sets in a single download, do not specify this attribute.

Type: date

Required: No-default is just before the beginning of time (negative infinity).

expiry

For the date-based validity option, the current date must precede this date for the data set to be valid.

Type: date

Required: No-default is just after the end of time (positive infinity).

period

For the rolling-period validity option, this attribute specifies the length of the periodic interval in terms of units specified with the *periodunit* attribute below. It must equal a non-negative integer.

Type: number

Required: Yes, but only if the *validitytype* attribute equals *rolling-period*.

periodunit

For the rolling-period validity option, this attribute specifies the units for the periodic interval: years, months, and so on.

Type: keyword-valid values are year, quarter, month, week, and day.

Required: Yes, but only if the *validitytype* attribute equals *rolling-period*.

dependson

For all data sets in a data set family except the parent data set, this attribute specifies the name of the parent data set in the family. For example, all incremental data sets will have this field set to the name of the full data set.

Type: String

Required: No

syncdefaultfirsttime

True if the data set should be downloaded by default for the first sync of the data set family on a given mobile (laptop) machine, false otherwise. Although sophisticated end users may wish to use the advanced area of the Disconnected Analytics Application Manager to specify precisely which data sets to download for a given sync operation, novice users may prefer to avoid the advanced area and take the default option.

Type: boolean keyword-true or false.

Required: No

syncdefaultsubsequenttimes

True if the data set should be downloaded by default for all subsequent sync operations (after the first synchronization) of the data set family on a given mobile machine, false otherwise. Again, sophisticated end users may wish to use the advanced area of the Disconnected Analytics Application Manager to specify precisely which data sets to download for a given sync operation, but novice users may prefer to avoid the advanced area and take the default option.

Type: boolean keyword-true or false. (yes and no may be supported later.)

Required: No

syncmode

Synchronization can take place in one of two ways: preprocessed or online. For the online option, Siebel Business Analytics Web on the enterprise server will run a sourcing report to generate the data files for the data set as part of the sync operation. Administrators typically use the online option for real-time data.

For the pre-processed option, Siebel Business Analytics Web on the enterprise server will look for existing data files to download and will only run the sourcing report if no such preprocessed files exist.

Type: keyword-valid values are pre-processed and online.

Required: No

forcesync

The advanced mode of the Disconnected Analytics Application Manager allows users to choose not to download a particular data set. The *forcesync* attribute allows the administrator to override that choice, forcing the Disconnected Analytics Application Manager to perform the download. Administrators typically use this attribute for system-related data and for commonly-used dimension tables such as a time dimension.

Type: boolean keyword-true or false. (yes and no may be supported later.)

Required: No

subjectarea

The name of the subject area in the enterprise repository associated with the mobile application. Siebel Business Analytics Web on the enterprise server uses this name when storing filters for the application in the enterprise Web Catalog.

Type: String

Required: Yes

■ **XML Tag:** <displayname>

This identifies the descriptive name of either an application or a data set in a specified language. The descriptive name for an application appears in the Disconnected page of Siebel Business Analytics Web on the enterprise server and in the application pull-down menu of the Disconnected Analytics Application Manager on the mobile machine. The descriptive names for all data sets appear in the advanced section of the Disconnected Analytics Application Manager and on the Disconnected Filter page of Siebel Business Analytics Web.

An application or data set may have multiple <displayname> tags each corresponding to a different language. Users logging in to Siebel Business Analytics Web specify their choice of language. Based on this choice, the corresponding display name will appear.

If a data set contains no <displayname> tag (or if a user specifies a language for which no <displayname> tag exists), the data set's internal name appears in the advanced section of the Disconnected Analytics Application Manager.

If an application contains no <displayname> tag (or if a user specifies a language for which no <displayname> tag exists), the application's default display name appears instead.

NOTE: For applications, the <displayname> tag must follow the <webcatalog> tag and the <postdbschema> tag (if specified); for data sets, the <displayname> tag must precede the first <table> tag.

Parent tag: <application> or <dataset>

Attributes:***lang***

(A code that represents the display name language.) Oracle's Siebel Mobile Analytics currently supports the following languages and their corresponding codes:

□ Czech	"cs"
□ Danish	"da"
□ German	"de"
□ English	"en"
□ Spanish	"es"
□ Finnish	"fi"
□ French	"fr"
□ Italian	"it"
□ Japanese	"ja"
□ Korean	"ko"
□ Dutch	"nl"
□ Portuguese	"pt"
□ Brazilian Portuguese	"pt-br"
□ Swedish	"sv"
□ Chinese	"zh"
□ Chinese Traditional	"zh-tw"

Type: String (code for the display name language).

Required: Yes

value

The actual text of the display name in the appropriate language.

Type: String

Required: Yes

■ **XML Tag:** <filterables>

An administrator may optionally use this tag to enumerate a set of columns from the sourcing report that end users can specify in filters to restrict what rows get downloaded to the table on the mobile (laptop) machine.

Parent tag: <sourcingreport>

Attributes: None

■ **XML Tag:** <formula>

This tag identifies a particular column from the sourcing report as available for use in filters. For example, in the Retail database described in [“Scenario for Using SQL Scripts to Create Mobile Tables and Indexes” on page 182](#), the Store sourcing report has columns such as Store.Name (with values such as Downtown and West End) and Store.city (with values such as San Francisco and Seattle). Administrators wanting to make these columns available for filtering would specify the following tags:

```
<sourcingreport name = "Store" file = "Store.csv">
```

```
<filterables>
```

```
<formula>Store.Name</formula>
```

```
<formula>Store.City</formula>
```

```
</filterables>
```

```
</sourcingreport>
```

End users could then add more filters to the sourcing report. For example, they could add a filter to download only the Store records for San Francisco and Seattle.

Alternatively, administrators can supply * as the column name. This indicates that all filterable columns previously specified in the data set apply to the sourcing report. For example, suppose the SalesFact sourcing report has the following filter specified:

```
<sourcingreport name = "SalesFact" file = "SalesFact.csv">
```

```
<filterables>
```

```
<formula>*</formula>
```

```
</filterables>
```

```
</sourcingreport>
```

This means that all filterable columns on previous sourcing reports (Store.Name and Store.color in our example), also apply to the SalesFact sourcing report. Therefore, if an end user adds filters to download only the Store records for San Francisco and Seattle, then only the fact table records for San Francisco and Seattle will be downloaded as well.

Note that administrators can only specify the * option for the last sourcing report in a data set.

Parent tag: <filterables>

Attributes: none

■ XML Tag: <indexsql>

Parent tag: <table>

Attributes:

name

The name of a SQL script file that the Disconnected Analytics Application Manager will use for creating one or more indexes on a table downloaded to the mobile (laptop) machine.

Type: String

Required: Yes

■ XML Tag: <messages>

Parent tag: <application>

Attributes:

dir

The name of the mobile application message directory described in [“Mobile Application Message Directory” on page 181](#). This directory contains the message localization XML files for the mobile Web Catalog of the application.

Type: String

Required: Yes

■ XML Tag: <postdbschema>

The <postdbschema> tag is optional, but if specified, it must follow the <webcatalog> tag.

Parent tag: <application>

Attributes:

name

The name of a SQL script file that the Disconnected Analytics Application Manager will run once it has loaded all data for the application into the SQL Anywhere database on the mobile (laptop) machine. Administrators might set up such a script file for updating statistics or reorganizing the database.

Type: String

Required: Yes

■ XML Tag: <remotecfg>

This tag brackets the entire XML file, identifying it as a mobile application configuration file.

Parent tag: none

Attributes: none

■ **XML Tag:** <repository>

The <repository> tag is mandatory and must precede the <webcatalog> tag.

Parent tag: <application>

Attributes:

name

The name of the mobile application repository. Oracle's Siebel Business Analytics Server on the mobile machine uses this repository as its source for Siebel Business Analytics metadata.

Type: String

Required: Yes

■ **XML Tag:** <sourcingreport>

Parent tag: <table>

Attributes:

name

The name of the Siebel Business Analytics Web report that generates the data for a particular table on the mobile (laptop) machine.

Type: String

Required: No, not necessary for externally generated data files.

file

The name of the output file generated from the sourcing report, or alternatively, it could be the name of an externally generated file (for example, syndicated data). This file is sent to the mobile machine and used to load the table. This attribute need not be specified; by default the output file name is the same as the name attribute followed by a CSV suffix.

Type: String

Required: Yes, only if the name attribute is not specified.

■ **XML Tag:** <table>

Parent tag: <dataset>

Attributes:

name

The name of the table stored in the SQL Anywhere database on the mobile (laptop) machine.

Type: String

Required: Yes

■ XML Tag: <tablesql>

Parent tag: <table>

Attributes:

name

The name of the script file that the Disconnected Analytics Application Manager will use for creating or updating the relevant table on the mobile (laptop) machine. For full data sets, the script file will typically contain a DROP TABLE statement followed by a CREATE TABLE statement. For incremental data sets, the table already exists on the mobile machine, so the script file will typically contain DELETE statements followed (optionally) by INSERT statements.

Type: String

Required: Yes

■ XML Tag: <webcatalog>

The <webcatalog> tag is mandatory and must follow the <repository> tag.

Parent tag: <application>

Attributes:

name

The name of the mobile application Web Catalog. Oracle's mobile Siebel Business Analytics Web Server uses this Web Catalog to store dashboards and reports.

Type: String

Required: Yes

Index

A

Abs math function, about 394

Acos math function, about 394

Administration Tool

See also Browse dialog box; logical business models; repository mode

Analytics Server, using to shut down 242

Cache Manager, changing column order 35

Cache Manager, selecting columns to display 35

Edit menu, described 22

File menu, described 22

Help menu, described 23

icons and symbols (table) 25

join diagrams, setting default window size 35

keyboard shortcuts (table) 24

main window, repository parts described 21

Manage menu, described and functions (table) 23

preferences, setting general preferences 32

repository components 51

repository objects, adding or editing 36

repository objects, setting permissions 35

repository objects, specifying appearance in alphabetical order 34

row counts, about updating for tables and columns 37

scrolling speed, setting 35

toolbar functions 24

Tools menu, described 23

View menu, described 22

Window menu, described 23

Administration Tool, using to shut down server 242

aggregate expression builder dialog

See Expression Builder dialog boxes

aggregate fact table

creating and example, about 112

aggregate functions

about 377

aggregate queries, about 370

alternative syntax 374

Avg aggregate function, about 377

AvgDistinct, calculating average mean 377

baseline columns, computing aggregates 370

BottomN, about ranking the lowest n values 377

Count (*) (CountStar), about counting number of rows 378

Count, about calculating the number of rows 378

CountDistinct, about adding distinct process to COUNT 378

display function, reset behavior 373

First, about selecting first returned value 379

GroupByColumn, about specifying logical columns 379

GroupByLevel, about setting up aggregate navigation 379

LAST, about selecting last returned value 379

Max, about calculating maximum value 380
measure columns, computing aggregates 371

Median, about calculating the median value of rows 380

Min, about calculating the minimum value 380

NTILE, about determining the rank of a value 381

Percentile, about calculating a percent rank 381

Rank, about calculating the rank for each value 382

StdDev, about returning standard deviation 382

Sum, about calculating 382

SumDistinct, about calculating sum by adding distinct values 383

TopN, about ranking the highest n values 383

aggregate navigation, setting up

See also aggregate tables; aggregate table fragments

aggregate levels, specifying for each source 227

aggregate table definitions, about and navigation 51

aggregate table fragments, about and example 232

aggregate tables, about 227

- aggregated fact data, creating dimension sources for each level 112
 - fragment content, specifying 227
 - WHERE clause filter, about and example 119
 - aggregate table fragments**
 - about and example 232
 - about configuring a repository to use fragments 232
 - aggregate table content, specifying 233
 - physical joins for virtual table, constructing 235
 - physical layer table, defining with a Select Statement 233
 - physical layer table, example 234
 - SQL virtual table content, creating 234
 - aggregate tables**
 - See also* aggregate table fragments
 - about and navigation 227
 - aggregate table definitions, about and navigation 51
 - performance, about using to improve 255
 - aliases**
 - Alias tab, using 141
 - synchronizing 214
 - Allow direct database requests for all, overriding (procedure)** 343
 - Allow populate queries for all, overriding (procedure)** 343
 - Analytics Event Tables utility, about and starting** 215
 - Analytics Server**
 - See also* Cluster Server feature
 - automatic startup, configuring for 238
 - connecting to, about 242
 - fails to start 240
 - nonlocal files, about accessing 348
 - note, about changing repository in online mode and attempting to stop server 32
 - password, changing 341
 - repository variable initialization blocks, re-initializing 311
 - starting 237
 - UNIX, running the shutdown script 241
 - UNIX, running the Siebel startup script 238
 - user ID, changing 239
 - user IDs and passwords, about and storage of 341
 - Windows, configuring for automatic startup 238
 - Windows, shutting down from a Windows command prompt 241
 - Windows, shutting down from Services 240
 - Windows, starting from Services 237
 - Analytics Server Administrator**
 - account, about and password 329
 - group, about 330
 - Analytics Server XML Gateway**
 - See* XML Gateway
 - analyzing the cache**
 - See* Cache Manager
 - ASCII string function, about** 388
 - Asin math function, about** 395
 - Ask DBMS button, using to change Feature table entries** 65
 - Atan math function, about** 395
 - Atan2 math function, about** 395
 - authentication cache, about disabling** 335
 - authentication options**
 - See also* security; groups, working with
 - Analytics Server internal authentication, about 341
 - Analytics Server user IDs and passwords, about and storage of 341
 - authentication, about 336
 - authentication, order of 342
 - database authentication, about using and procedure 340
 - external table authentication, about 338
 - external table authentication, setting up 338
 - LDAP authentication, about 337
 - LDAP authentication, setting up 337
 - logging level, setting 338
 - password, changing 341
 - security, bypassing 342
 - USER session system variable, defining for LDAP authentication 337
 - Avg aggregate function, about** 377
 - AvgDistinct aggregate function, about** 377
- B**
- baseline column**
 - behavior with aggregate functions 370
 - example 370
 - Between SQL logical operator, about** 374
 - Bit_Length string function, about** 388
 - BottomN aggregate function, about** 377
 - Browse dialog box**
 - about using 38
 - querying for an object 39
 - selecting an object in 39
 - synchronizing an object in the query results list with the tree control list 39
 - buffer size, configuring** 261
 - business model**
 - definition 43
 - Business Model and Mapping layer**

- about creating and maintaining 103
- Business Model and Mapping layer, setting up**
 - logical table sources (mappings), setting up 111
 - table sources content definitions, creating 116
- Business Model and Mapping layer, working in**
 - See also* business models; logical table joins
 - business model, about working with 104
 - business model, creating (procedure) 104
 - logical columns, creating 107
 - logical table source 111
 - logical tables, about working with 105
 - measure, associating with a level in dimension 110
 - measure, removing the association 110
 - physical to logical mapping, defining 113
 - repository, about setting up and working in 53
- business models**
 - See also* Business Model and Mapping layer, working in; presentation catalogs
 - consistency check, about passing 30
 - copy business model with presentation catalog utility 104
 - diagram, about using to create joins 129
 - diagram, displaying 131
 - repository, checking consistency within a 31
 - understanding 42

C

cache

- behavior in offline mode 273
- behavior in online mode 273
- disabling 262
- enabling 262
- purging 275
- purging when switching between repositories 273

cache authentication, about disabling 335

cache hits

- description of 265
- information about 265

cache information, viewing 321

Cache Manager

- about and opening 274
- column order, changing 35
- columns, selecting columns to display 35
- global cache information, displaying 275
- option settings (table) 274
- purging cache 275

- view and save SQL call 274

cache persistence time setting 263

cache storage

- cache data storage directories 261
- cache entry, controlling max number of rows 261
- cache metadata file, about 261
- query caching, disabling query caching 262
- query caching, enabling query caching 262

Cache, Manage menu option, described 23

cache, monitoring and managing

- caching attributes, setting for physical tables (procedure) 263
- disabling caching for the system, about 262
- event polling tables, configuring 263
- physical tables, about caching and persistence timing 263

Calculation Wizard, about 33

calendar date/time functions

- Current_Date, about returning 400
- Current_Time, about 400
- Current_TimeStamp, about 401
- Day_Of_Quarter, about 401
- DayName, about 401
- DayOfMonth, about 401
- DayOfWeek, about 402
- DayOfYear, about 402
- Hour, about 402
- Minute, about 402
- Month, about 403
- Month_Of_Quarter, about 403
- MonthName, about 403
- Now, about 403
- Quarter_Of_Year, about 404
- Second, about 404
- TimeStampAdd, about 404
- TimeStampDiff, about 406
- Week_Of_Quarter, about 407
- Week_Of_Year, about 407
- Year, about 408

Cast conversion function, about 408

catalog folders

- creating or editing a catalog 93
- Dynamic Name tab, sorting entries 94
- Dynamic Name tab, specifying the session variable to use 93
- Dynamic Name tab, unassigning a session variable 94

Ceiling math function, about 395

Char string function, about 388

Char_Length string function, about 389

character literals, about 410

check global consistency

- options when saving 55

checking in changes

- changes, making available and saving to disk 32
- Check In Changes dialog box, about using and tasks 31

checkpoints

- described 252

client tools, connectivity to 281**cluster information, viewing**

- cache information, viewing 321
- Cache view, about and columns described (table) 320
- Request window (Session view), columns described (table) 322
- Session window (Session view), columns described (table) 322
- Session window, columns described (table) 321
- Status view, about and columns described (table) 319

Cluster Manager

- See also* Cluster Server feature
- accessing (procedure) 318
- cache information, viewing 321
- Cache view, about and columns described (table) 320
- graphical user interface, described 318
- managing clustered servers (procedure) 324
- managing clustered servers, actions available 324
- performance considerations 324
- refreshing, setting and procedure 319
- Session view, Request window (Session view) columns described (table) 322
- Session view, Session and Request window, described 321
- Session view, Session window columns described (table) 322
- Status view, about and columns described (table) 319
- stopped or offline, note about starting 318
- using, about 318

Cluster Server feature

- See also* Cluster Manager
- about 313
- analytics ODBC data source name, about configuring 316
- cache information, viewing 321
- Cluster Controller and Analytic Server, starting manually in Windows 316
- Cluster Controller and Analytics Server, starting from the Command window 316
- installing 315

- managing clustered servers (procedure) 324
- managing clustered servers, actions available 324
- master server, about 314
- NSClusterConfig.INI file, about copying to Config directory 316
- parameters, setting in the NQClusterConfig.INI file 315
- parameters, setting in the NQSCConfig.INI file 315
- performance considerations 324
- primary Cluster Controller, about 313
- Repository Publishing Directory, about 314
- secondary Cluster Controller, about 313
- startup process overview 317
- stopped or offline, note about starting 318

Cluster, Manage menu option, described 23**column mapping**

- logical columns to physical columns, mapping 113
- removing column mapping 114

Command window, stating Cluster Controller and Analytics Server 316**complex joins**

- about 96
- logical complex joins, about and complex joins 129
- logical complex joins, creating 131

Concat string function, about 389**conditional expressions**

- CASE (if), about and syntax 375
- CASE (Switch), about and syntax 374

connection pool

- about 52
- creating and configuring, about 65
- creating or editing 68
- dialog box, fields described 71
- persist connection pool property, setting up 79
- persist connection pool, assigning (procedure) 80

connectivity

- client tools and data sources, connectivity to 281
- metadata, importing 281
- ODBC data source names, configuring 279
- query and reporting tools, about using 282

consistency check

- about passing and example 30
- business model within a repository, checking consistency of 31
- repository, checking for consistency 31

conversion functions

- Cast, about changing the data type to another

- data type 408
- IfNull, about testing if an expression evaluates to a null value 409
- VALUEOF(), about using the function in an expression builder 409
- copy business model with presentation catalog utility, about and procedure** 104
- Cos math function, about** 396
- Cot math function, about** 396
- Count (*)/CountStar aggregate function, about** 378
- Count aggregate function, about** 378
- CountDistinct aggregate function, about** 378
- cube metadata**
 - alias-SQL file, executing 292
 - alias-SQL output file, about 285
 - deploying 292
 - import files, creating 283
 - input file, about 284
 - materialized query tables (MQTs), guidelines 294
 - objects, mapping, about 286
 - output files, about 285
 - XML file, importing 292
 - XML output file, about 285
- CubeViews Generator**
 - about 282
 - alias-SQL output file, about 285
 - analytics repository metadata objects, list of 286
 - cube metadata input file, about 284
 - errors, list of 285
 - IBM Cube Views metadata objects, list of 286
 - import files, creating 283
 - metadata objects, conversion rules for 287
 - metadata objects, mapping, about 286
 - metadata objects, validation rules for 287
 - output files, about 285
 - running 283
 - SACubeViewsGen.exe, executing 283
 - SACubeViewsGen.exe, parameters of 284
 - troubleshooting 285
 - XML output files, about 285
- Current_Date calendar date/time function, about** 400
- Current_Time calendar date/time function, about** 400
- Current_TimeStamp calendar date/time function, about** 401

D

Data Mining Adapter

See also XML Gateway; XML ODBC database type; XML, using as a data source

- configuring (procedure) 360
- In-Process Data Mining Adapter API, about 358
- In-Process Data Mining Adapter API, sample implementation 359
- In-Process Data Mining Adapter API, using ValueOf() expression 359
- operation modes 357

data modeling

- business model, understanding 42
- objectives of 42

data sources

- connectivity to 281

DATA_STORAGE_PATHS parameter, using to specify query cache storage

database authentication

- Delivers, about 340

database hints

See also databases

- database objects that accept objects (table) 99
- hints, creating (procedure) 100
- index hint, about 100
- Leading hint, about 100
- performance considerations 100
- SQL comment markers, about entering 101
- usage examples 99
- using, about 99

database object

- creating manually in the Physical layer 62
- database hints, database objects that accept hints (table) 99
- ODBC type, about assigning if database type undetermined 62

Database system function, about

database type, restoring default entries for

Database types

- multidimensional data source, automatically assigning 62
- relational data source, automatically assigning 62

databases

See also database hints

- authentication, about using and procedure 340
- configuring, tuning, and indexing, importance 256
- database hints, about using 99

- symbol, described 25
 - datetime literals, about** 411
 - Day_Of_Quarter** calendar date/time function, about 401
 - DayName** calendar date/time function, about 401
 - DayOfMonth** calendar date/time function, about 401
 - DayOfWeek** calendar date/time function, about 402
 - DayOfYear** calendar date/time function, about 402
 - DCOM, changed client/server communication method** 280
 - decimal literal, about** 411
 - default client/server communication method, changed from DCOM to TCP/IP** 280
 - Default initializer column, about value** 311
 - Degrees** math function, about 396
 - deleting**
 - alias 141
 - column mapping 114
 - initialization block 311
 - measure, removing the association 110
 - presentation tables 139
 - presentation tables column (procedure) 140
 - table as a logical table source 112
 - description of** 265
 - dimensional hierarchies**
 - measures, about 43
 - sample hierarchy rollups, about and diagram 45
 - star and snowflake models, about 46
 - dimensional hierarchy**
 - grand total levels, about 120
 - grand total levels, example 125
 - grand total, example 125
 - level attributes, about 120
 - level-based measure calculations, setting up 124
 - level keys, about 121
 - level-based measure calculations, about 123
 - level-based measure calculations, example 123
 - levels, about creating 120
 - dimensional level**
 - general properties, defining 121
 - primary key, adding 123
 - dimensional models**
 - about 47
 - sample hierarchy rollups, about and diagram 45
 - single table model, about and creating 55
 - understanding 43
 - dimensional schemas**
 - about and advantages 48
 - fact table, about 49
 - star schema 48
 - dimensions**
 - about 119
 - about cubes from a multidimensional data source 119
 - creating (procedure) 120
 - creating and administering 119
 - hierarchies, about 119
 - note, about including the key column in the dimension 120
 - dimensions, defined and example** 44
 - dimension-specific aggregation rules**
 - columns, specifying for 127
 - setting up, about 127
 - display functions**
 - example 373
 - reset behavior 373
 - DISPLAYNAME** system session variable, about 300
 - dragging and dropping**
 - logical tables 106
 - driving table**
 - about specifying 132
 - caution, about specifying when creating a Business Model Diagram 132
 - caution, about specifying when creating a logical complex join 131
 - caution, about specifying when creating a logical foreign key 130
 - caution, specifying and query optimization 132
 - controlling and tuning performance, about 133
 - DSN connection**
 - See Analytics Server
 - Dynamic Name** tab
 - entries, sorting 94
 - session variable, specifying 93
 - session variable, unassigning 94
 - dynamic repository variables**
 - See also initialization blocks
 - about 298
 - example 299
 - initializing 304
- ## E
- Edit menu, described** 22
 - EMAIL** system session variable, about 301
 - entity-relationship (E-R) models**

about 47, 48
queries that perform historical analysis,
performance of 48

event polling table

Analytics Event Table utility, identifying
using 215
cache event processing, about 268
configuring 263
CREATE TABLE statements, sample event
polling table 270
making the polling table active 271
overview 263
physical databases, setting up 268
populating, about 272
repository, making changes to 273
troubleshooting 272
using 273

Exp math function, about 396

**Export logical keys option, about using with
parameterized SQL queries** 138

Expression Builder dialog boxes

about using 219
accessing 219
Aggregate Content folder, about 221
Constraints folder, about 222
Dimensions folder, about 222
example (diagram) 220
expression, building (procedure) 224
Expressions folder, about 222
Functions folder, about 222
Logical Tables folder, about 222
navigating within the expression builder 224
Operators folder, about 222
Repository Variables folder, about 222
Session Variables folder, about 222
setting up example 223
toolbar (table) 221
Types folder, about 222

expression literals

character literals, about and expressing 410
datetime literals, about and formats 411
decimal, about and expressing 411
floating point, about and expressing 411
integers, about and expressing 411
numeric literals, about 411

Extensible Markup Language

See XML, using as a data source

external table authentication

about 338
setting up 338

**Externalize Strings utility, about and
starting** 215

F

**Feature table entries, changing using Ask
DBMS** 65

File menu, described 22

filter

See also query execution privileges
complex filter, about constructing 151
constructing a filter to view all databases
references in a business model 150
constructing a filter to view all Presentation
layer columns mapped to a logical
column 151
query results, constructing a filter for 150

First aggregate function, about 379

floating point literal, about 411

Floor math function, about 397

folder symbol, described 25

foreign keys

logical foreign key, creating 130
note, about importing from an Oracle
database 58
primary key, relationship with 96

fragmentation content

about 227
multicolumn content descriptions
example 228
parallel content descriptions example 229
single column range-based predicates
example 228
single column value-based predicates
example 227
unbalanced parallel content descriptions
example 231

fragmented data, about and example 95

FROM clause syntax, about 369

functions

See aggregate functions

G

grand total dimension hierarchy

example 125

grand total levels

about 120

example 125

granularity, defined 49

graphical user interface

See Cache Manager

GROUP BY clause

query behavior with and without 373
syntax, about 369

GROUP system session variable, about 300

**GroupByColumn aggregate function,
about** 379

GroupByLevel aggregate function, about specifying dimension levels 379
groups, working with

See also authentication options
 about 329
 groups, about creating and example 330
 LDAP authentication, configuring 334
 LDAP, about importing users and groups 333
 member hierarchies, viewing in the Query Repository dialog box 333
 member hierarchies, viewing in the Security Manager 333
 predefined Administrators group, about 330
 privileges example (diagram) 332
 privileges, about granting and examples 331
 repository, adding a new group to 332
 repository, importing LDAP users and groups into 336
 user privileges and group privileges example (diagram) 331

H

Help menu, described 23

HIDD_A_KEY 88

hierarchies

about 119

hierarchies, about dimensional hierarchies 43

hierarchy, defined and example 45

historical time comparison, about and starting Times Series Wizard 213

Hour calendar date/time function, about 402

HTML tables

example 356

XML Gateway, accessing by 356

I

IBM Cube Views

about 282

alias-SQL file, executing 292

deploying cube metadata 292

materialized query tables (MQTs), guidelines 294

metadata

objects, list of 286

XML file, importing 292

ibot, definition 166

icons, described (table) 25

IfNull conversion function, about 409

importing

metadata 281

metadata, about connecting using an ODBC

connection 59

users and groups using LDAP 333

XML data using ODBC 361

In SQL logical operator, about 374

inconsistencies, about checking repository or business model for consistencies 30

indexing, about index hint instructions 100

INI file

NQClusterConfig.INI file, setting parameters for Cluster Server feature 315

NQSCConfig.INI file, tuning parameters 255

NSQConfig.INI file, adding an entry 144

Initialization Block dialog box

accessing (procedure) 310

using, about 310

initialization blocks

about 302

Analytics Server, re-initializing when it starts 311

block, linking with a variable not already refreshed 310

block, removing variable's association with 311

blocks, removing 311

caution, about opening Initialization Block dialog box in online mode 303

Default initializer column, about value 311

dynamic repository variables, initializing 304

initialization block execution order, setting 311

new variable, adding to refresh with block 310

note, about number of columns different from number retrieved 310

session variables, creating or editing (procedure) 306

session variables, initializing 304

variable refreshed by this block, editing

variable, reordering 310

In-Process Data Mining Adapter API

about 358

column values, specifying 359

configuring (procedure) 360

sample implementation 359

ValueOf() expressions, using 359

Insert string function, about 389

integers literals, about 411

Is Null SQL logical operator, about 374

J

Jobs, Manage menu option, described 23

join diagrams, setting default window

size 35

Joins Manager
joins, about using to create 129

Joins, Manage menu option, described 23

K

key symbol, described 25, 26

keyboard shortcuts (table) 24

L

Last aggregate function, about 379

LDAP
See Lightweight Directory Access Protocol (LDAP)

Leading hint, about 100

Left string function, about 390

Length string function, about 390

level attributes
about 120

level keys
about 121

level-based measure calculations
about 123
example 123
setting up 124

levels
general properties, defining 121
hierarchy, about 121
primary key for, adding 123

levels, working with
grand total levels, example 125
level-based measure calculations, setting up 124

Lightweight Directory Access Protocol (LDAP)
authentication, about 337
authentication, setting up 337
LDAP authentication, configuring 334
logging level, setting 338
passwords and storage, about 341
repository, importing LDAP users and groups into 336
USER session system variable, defining for LDAP authentication 337
users and groups, about using to import 333

Like SQL logical operator, about 374

literals, expressing (list of) 410

Load all objects option, about selecting 30

Locate string function, about 390

LocateN string function, about 391

log files
See also query log, administering; usage tracking, administering

NQServer.log and NQCluster.log, about opening and examining 317

See log files

Log math function, about 397

log viewer utility

log records, interpreting 245

running (procedure) 245

using, about 245

Log10 math function, about 397

logging levels

individual users, enabling 243

levels described (table) 244

log viewer utility, interpreting the log records 245

log viewer utility, using 245

override with session variable 243

user's logging levels

disabling 245

setting 244

logical business models

bridge tables, identifying 45

dimensional models, understanding 43

single table model, about and creating 55

Logical Column dialog box

measure, associating with a level in a dimension 110

measure, removing the association 110

logical columns

creating or editing, about 108

creating, about 107

creating, procedure 108

logical column, unmapping from its source 114

logical complex join, creating 131

logical foreign key, creating 130

logical object, displaying all that map to physical tables 133

Logical Table dialog box

foreign key, editing 107

key, specifying 106

new logical table source, adding 111

logical table joins

See also Business Model and Mapping layer, working in

about 129

Business Model Diagram, displaying 131

creating, about 129

driving table, about specifying 132

driving table, about specifying and query optimization 132

driving table, controlling and tuning performance 133

logical complex join, creating 131

- logical foreign key, creating 130
 - logical object, displaying physical tables that map to 133
 - logical table sources**
 - removing a table as a source 112
 - Replace Wizard, using to replace tables or columns 214
 - setting up, about 111
 - Where clause filter, using to constrain physical tables 119
 - logical table sources, editing** 106
 - logical tables**
 - creating by dragging and dropping 106
 - creating explicitly 106
 - creating, ways to 105
 - foreign key, editing 107
 - key, specifying 106
 - new logical table source, adding 111
 - working with, about 105
 - LOGLEVEL system session variable, about** 301
 - Lower string function, about** 391
- ## M
- main window, repository parts described** 21
 - Manage menu, described and functions (table)** 23
 - Marketing, Manage menu option, described** 23
 - MASTER_SERVER parameter, about** 314
 - math functions, list of** 394
 - Mavg running aggregate function, about** 384
 - Max aggregate function, about** 380
 - measure**
 - dimension, associating with 110
 - dimension, removing the association 110
 - measure column**
 - behavior with aggregate functions 371
 - default aggregation rule, specifying a 109
 - example 372
 - measure column, specifying a default aggregation rule** 109
 - Median aggregate function, about** 380
 - member hierarchies**
 - Query Repository dialog box, using to view 333
 - Security Manager, viewing in 333
 - metadata, importing**
 - about 281
 - connections, about 59
 - Min aggregate function, about** 380
 - Minute calendar date/time function, about** 402
 - Mobile Analytics**
 - Disconnected Analytics Application Manager utility, about 177
 - Mod math function, about** 397
 - modes**
 - See offline mode; online mode
 - Month calendar date/time function, about** 403
 - Month_Of_Quarter calendar date/time function, about** 403
 - MonthName calendar date/time function, about** 403
 - More tab**
 - joins diagrams, using to set default window size 35
 - scrolling speed, using to set 35
 - MSUM running aggregate function, about** 384
 - multi-database joins, about** 95
 - multidimensional**
 - dimensions for a cube from a multidimensional data source 119
 - multidimensional data source**
 - Ask DBMS, about availability of 65
 - physical layer, creating 60
 - physical schema, importing 61
- ## N
- Now calendar date/time function, about** 403
 - NQClusterConfig.INI file, setting parameters for Cluster Server feature** 315
 - nQLogViewer utility**
 - log records, interpreting 245
 - query log file, using to view 245
 - NQSCConfig.INI file**
 - entry, adding an 144
 - tuning parameters, about 255
 - NQServer.log, and NQCluster.log, about opening and examining** 317
 - NTile aggregate function, about** 381
 - numeric literals, about** 411
- ## O
- Object Type option, using to create virtual physical tables** 81
 - Octet_Length string function, about** 391
 - ODBC**
 - calls from client applications, list of 295
 - client tools and data sources, about providing connectivity to 281
 - connection, about physical metadata

- imports 59
- data source names, configuring 279
- metadata, importing 281
- query and reporting tools, about connecting with 282
- offline mode**
 - cache invalidation, implications for 273
 - repository, opening in 29
- one-to-many relationships, about primary key-foreign key relationship (diagram)** 50
- online help, accessing** 29, 51
- online mode**
 - cache invalidation, implications for 273
 - changes, making available and saving to disk 32
 - Check In Changes dialog box, about using and tasks 31
 - consistency check, about passing 30
 - consistency, checking repository for 31
 - repository, opening in 30
- opaque view**
 - best practice, analytics repository 53
 - definition 53
- Options dialog box, using to set general preferences** 32
- Oracle database**
 - note, about importing foreign keys from 58
- Oracle database, stored procedures, using** 86
- ORDER BY clause, about** 370

P

- parallel content descriptions, examples and discussion** 229
- password**
 - Analytics Server administration account 329
 - changing 341
- Percentile aggregate function, about** 381
- performance**
 - database hints, about resulting in better query performance 100
 - server configuration and tuning 255
- permissions**
 - See also* privileges
 - adding or editing 36
 - limiting queries by filtering (procedure) 345
 - limiting queries by maximum run time (procedure) 344
 - limiting queries by number of rows received (procedure) 344, 346
 - limiting queries by objects (procedure) 343
 - limiting queries by time periods

- (procedure) 344
- repository objects, about setting for 35
- physical column, creating or editing** 86
- physical database model, understanding**
 - aggregate navigation, about setting up 51
 - primary-key-foreign key relationships, about and diagram 50
- physical databases**
 - analyzing 50
 - dimensional schemas, about and advantages 48
 - dimensional schemas, about star schema 48
 - entity-relationship (E-R) schemas, about 48
 - entity-relationship (E-R) schemas, performance for queries for historical analysis 48
 - identifying contents 50
 - types of 47
 - understanding 47
- Physical Diagram**
 - command, about using 133
 - displaying 98
 - editor, about using to specify multi-database joins 95
 - foreign key join or complex join, defining 98
 - physical joins, about defining 97
- physical joins**
 - See also* Physical layer, working in about 95
 - complex joins, about 96
 - fragmented data, about and example 95
 - Joins Manager, using to define 97
 - multi-database joins, about 95
 - note, about imported key and foreign key joins 95
 - primary key and foreign key relationships, about 96
 - tip, about avoiding unnecessary joins 96
- physical joins, defining in the Physical Diagram** 97
- Physical layer**
 - creating and maintaining, about 57
 - queries, specifying types of sent to a database 64
- Physical layer, creating manually** 62
- Physical layer, described** 52
- Physical layer, working in**
 - See also* physical joins
 - catalog, creating or editing 93
 - column mapping, removing 114
 - connection pool, creating or editing 68
 - database hints, about 99
 - database hints, database objects that accept hints (table) 99

- database type, restoring default entries for 65
- Feature table entries, changing using Ask DBMS 65
- logical columns, mapping to physical columns 113
- physical joins, defining with the Joins Manager 97
- query type, locating 65
- schema folders, working with 93
- XML data source, setting properties for 77
- physical schemas**
 - importing from ODBC (procedure) 59
 - importing, about 58
- physical tables**
 - creating or editing 84
 - overview 81
 - Physical Table dialog box, completing Columns and Keys tabs 86
 - virtual physical tables, creating using the Object Type option 81
 - XML data source, setting properties for 92
- Pi math function, about** 397
- populate privilege**
 - overriding database property 344
- PORTALPATH system session variable, about** 300
- Position string function, about** 392
- Power math function, about** 398
- preferences**
 - Cache Manager, changing column order 35
 - Cache Manager, selecting columns to display 35
 - general preferences, setting 32
 - join diagrams, setting default window size 35
 - repository objects, specifying appearance in alphabetical order 34
 - scrolling speed, setting 35
- presentation catalogs**
 - Alias tab, using 141
 - caution, about moving columns into presentation catalog folders 138
 - copy business model with presentation catalog 104
 - creating (procedure) 137
 - Presentation Catalog dialog box, described 137
 - presentation tables, deleting 139
 - table, reordering in the Presentation layer 139
 - tables, sorting in alphanumeric order 139
 - working in, about 137
- presentation columns**
 - Alias tab, using 141
 - creating (procedure) 140
 - editing (procedure) 140
 - Presentation Column dialog box, described (table) 140
 - Repository Documentation utility, using to map to logical and physical columns 217
 - working with, about 139
- Presentation Layer**
 - nested folders in Siebel Answers 139
- Presentation layer**
 - about creating 135
 - about creating and maintaining 135
- Presentation Layer dialog box, using the Alias tab** 141
- Presentation layer, creating**
 - See also* Presentation layer, working in business models, copying to the Presentation layer 135
 - columns, about removing unneeded or unwanted 135
 - logical keys, about exporting in the Presentation Catalog 136
 - presentation columns, renaming 136
 - Presentation layer repository layer, about 55
- Presentation layer, setting up**
 - presentation column, creating 140
 - presentation column, deleting 140
 - presentation column, editing 140
- Presentation layer, working in**
 - See also* Presentation layer, creating
 - Alias tab, using 141
 - Presentation Catalog dialog box, described 137
 - presentation catalogs, about working in 137
 - Presentation Column dialog box, described (table) 140
 - presentation column, reordering 141
 - presentation columns, working with 139
 - Presentation layer, about and example 136
 - Presentation Tables dialog box, described (table) 138
 - presentation tables, creating 138
 - presentation tables, deleting 139
- presentation table, reordering** 139
- presentation tables**
 - Alias tab, using 141
 - column, deleting 140
 - creating (procedure) 138
 - presentation column, reordering 141
 - Presentation Tables dialog box, described (table) 138
- primary key**

foreign key, relationship with 96
specifying 88

primary key-foreign key relationship

about and diagram 50

PRIMARY_CONTROLLER parameter, about 313

privileges

query privileges, about controlling and
activities 343

Project, Manage menu option, described 23

Q

Quarter_Of_Year calendar date/time function, about 404

queries

aggregate functions, rules for 373
database, specifying types sent to 64
Leading hint, about using to build the join
order 100

query caching, enabling to improve
performance 256

query privileges, controlling and
activities 343

query type, locating 65

query caching

advantages of 258
cache event processing with an event polling
table 268
cache hits 265
cache hits, information about 265
cache management strategy, choosing 262
Cache Manager, about and opening 274
Cache Manager, options settings (table) 274
cache storage, configuring 261
cache strategies 265
cost of caching, about 259
disabling for system 262
disabling query caching 262
enabling query caching 262
global cache information, displaying 275
improve performance, enabling to 256
invalidation after offline repository
changes 273
note, about query references with different
persistence times 85
parameters to control query caching, location
of 258
purge cache options 275
purging cache 275
refresh interval, setting for XML data
sources 276
suite of queries, about running 267
user ids, initializing cache entries for 259

query environment, administering

See also server configuration and tuning;
usage tracking, administering

Analytics Server, shutting down in UNIX 241

Analytics Server, shutting down in
Windows 240

Analytics Server, starting in UNIX 238

Analytics Server, starting in Windows 237

query log, administering 242

server, about connecting to 242

usage tracking, administering 246

query execution privileges 343

query log, administering

about 242

file size, controlling 243

log viewer utility, interpreting the log
records 245

log viewer utility, using 245

logging levels

described (table) 244

setting a user's 244

logging system, configuring 243

user's logging levels

disabling 245

query log, setting logging levels 243

query repository

new object, creating (procedure) 150
procedure 148

Query Repository dialog box

about using 147

member hierarchies, using to view 333

parent of an object 149

searching for object based on name 149

Show Qualified Name 149

type of object 149

Query Repository Filter dialog box

about and accessing 150

filter, constructing 150

filter, constructing to view all database
references in a business model 150

query specification (SELECT statement), about 367

query tool, about connecting with 282

R

Radians math function, about 398

Rand math function, about 398

RandFromSeed math function, about 398

Rank aggregate function, about 382

RCOUNT running aggregate function, about 386

refresh interval, setting for XML data sources 276

Rename Wizard, about and starting 216

**Rename Wizard, using to rename
Presentation layer and Business
Model and Mapping layer tables and
columns** 216

Repeat string function, about 392

Replace string function, about 392

Replace Wizard, about and starting 214

**reporting tool, about connecting with
repositories** 282

- comparing repositories 151
- comparing, turning off Compare Mode 153
- LDAP authentication, configuring 334
- LDAP, importing users and groups into 336
- making changes to and implications for
cache 273
- merging repositories, about and process 153
- merging version of Analytics repository
(procedure) 156
- new group, adding to 332
- new user, adding to 328
- synchronizing 146
- synchronizing and updating (procedure) 146

repository

Siebel Delivers, setting up 166

**Repository Documentation utility, about and
starting** 217

repository file

- options when saving 55
- save options 55

**Repository Import Wizard, about and
using** 146

repository mode

- See also* individual repository entries
- business model within repository, checking
consistency of 31
- changes, making available and saving to
disk 32
- Check In Changes dialog box, about using and
tasks 31
- Load all objects option, about selecting 30
- note, editing while repository is being
loaded 29
- offline mode, opening repository in 29
- online mode, about passing consistency
check 30
- online mode, checking consistency of a
repository 31
- online mode, opening repository in 30

repository objects

- See also* individual repository entries
- alphabetical order, specifying in 34
- note, about naming objects Administrator 56
- permissions, adding or editing 36

permissions, setting 35

permissions, sorting columns 36

repository variables

See also individual repository entries and
initialization blocks

about 297

cache purging considerations 298

Default initializer column, about value 311

dynamic repository variables, about 298

dynamic repository variables, example 299

static repository variables, about 297

static repository variables, example 298

static repository variables, using in an
expression 298

static repository variables, using in expression
builder 298

uses for static repository variables 297

repository, managing metadata

See also individual repository entries and
Query Repository dialog box

complex filter, about constructing 151

note, about constructing more than one
filter 151

query results, constructing a filter for 150

repository, setting up

Administration Tools, repository components
in 51

Analytics Server, about starting 145

connection pool, about creating and
configuring 65

data source, about defining 145

new repository, creating 56

NQSSConfig.INI file, adding entry 144

online help, accessing 29, 51

physical schemas, about importing 58

saving 56, 104, 135

saving, checking consistency, and correcting
errors 143

testing and refining, about 145

user community, about publishing to 146

**REPOSITORY_PUBLISHING_DIRECTORY
parameter, about** 314

**REQUESTKEY system session variable,
about** 301

Right string function, about 392

**RMAX running aggregate function,
about** 386

**RMIN running aggregate function,
about** 387

rollover, described 252

Round math function, about 399

row count feature

updating for tables and columns 37

row counts

- displaying 38
- updating 38
- row counts in physical layer, displaying** 38
- Row-Wise Initialization feature**
 - about and example 304
 - variable with a List of Values, initializing 305
- RSUM running aggregate function, about** 385
- running aggregate functions**
 - about 383
 - Mavg, about calculating a moving average 384
 - MSUM, about calculating a moving sum 384
 - RCOUNT, about counting number of input records 386
 - RMAX, about showing the maximum values of a set of records 386
 - RMIN, about showing the minimum values based on a set of records 387
 - RSUM, about calculating a running sum 385
- S**
- SA system subject area**
 - about 166
- sample scripts, locating and example** 413
- schema folders**
 - schema object, creating 93
- schemas**
 - See also* schema folders
 - physical schemas, about importing 58
 - physical schemas, importing from ODBC (procedure) 59
- scrolling speed, setting** 35
- Second calendar date/time function, about** 404
- SECONDARY_CONTROLLER parameter, about** 314
- security**
 - See also* authentication options; privileges
 - Analytics Server administration account password 329
 - Analytics Server Administrator account, about 329
 - bypassing 342
 - group privileges, about granting and examples 331
 - groups, about creating and example 330
 - groups, about working with 329
 - LDAP authentication, configuring 334
 - LDAP, about importing users and groups 333
 - member hierarchies, viewing in the Query Repository dialog box 333
 - member hierarchies, viewing in the Security Manager 333
 - predefined administrators group 330
 - privileges example (diagram) 332
 - repository, adding a new group to 332
 - repository, adding new user to 328
 - repository, importing LDAP users and groups into 336
 - user accounts, about working with 328
 - user privileges and group privileges example (diagram) 331
- Security Manager**
 - See also* security
 - member hierarchies, using to view 333
- Security, Manage menu option, described** 23
- SELECT list syntax**
 - about and syntax 368
 - FROM clause syntax, about 369
 - GROUP BY clause syntax, about 369
 - ORDER BY clause syntax, about 370
 - WHERE clause syntax, about 369
- SELECT statement**
 - about and basic syntax 367
 - conditional expressions, list of 374
 - queries and aggregate functions, rules for 370
 - query specification 367
 - Select list syntax 368
 - SQL logical operators 374
 - usage notes 368
 - WHERE clause 369
- server configuration and tuning**
 - See also* query environment, administering
 - aggregate tables, about using 255
 - databases, importance of configuring, tuning, and indexing 256
 - NQSCfg.INI parameters. about using for tuning 255
 - query caching, about enabling to improve performance 256
- server session management**
 - See* server configuration and tuning; Session Manager
- Session Manager**
 - See also* query environment, administering
 - active query, killing 255
 - disconnecting a user from a session 254
 - Session Window fields (table) 254
 - session, viewing 254
 - update speed, controlling 253
 - using, about 253
- session variables**
 - See also* initialization blocks
 - about 297

- creating or editing (procedure) 306
 - initializing, about 304
 - nonsystem session variables, about
 - using 301
 - row-wise initialization, about and
 - example 304
 - system session variables, about using 300
 - system session variables, table of 300
 - using for authenticating users 299
- Sessions, Manage menu option, described** 23
- shutting down Analytics Server**
 - UNIX, shutting down server 241
 - Windows, shutting down from a command prompt 241
 - Windows, shutting down from Services 240
- Siebel Analytics Scheduler, about interface to** 23
- Sign math function, about** 399
- Sin math function, about** 399
- single table model, about and creating** 55
- SKIN system session variable, about** 301
- snowflake models, about dimensional hierarchies** 46
- Sort Objects tab, using to specify repository objects in alphabetical order** 34
- sorting columns**
 - alphabetical 108
 - chronological 108
 - lexicographical, defining 108
- sourcing query**
 - definition 176
 - standard query, difference between 176
- SQL 92 functions**
 - datetime literals, about 411
 - NTILE function, about 381
- SQL FROM clause syntax, about** 369
- SQL functions**
 - aggregate functions, about 377
 - calendar date/time functions, about 400
 - conversion functions, about 408
 - datetime literals, about 411
 - expression literals, about 410
 - NTILE function, about 381
 - running aggregate functions 383
 - string functions, about 388
 - system functions, about 410
- SQL logical operators, list of** 374
- SQL queries, about selecting the Export logical keys option** 136
- SQL statement**
 - database hints, about 99
 - database hints, creating 100
 - database objects that accept hints (table) 99
- SQL syntax and semantics**
 - conditional expressions, list of 374
 - queries and aggregate functions, rules for 370
 - Select list syntax 368
 - Select statement, about and basic syntax 367
 - Select usage notes 368
 - SQL logical operators 374
- SQL WHERE clause syntax, about** 369
- Sqrt math function, about** 399
- star schema**
 - about 48
 - dimensional hierarchies, about 46
- static repository variables**
 - See also* initialization blocks
 - about 297
 - example 298
 - expression builders, using in 298
 - expression, using in 298
- StdDev aggregate function, about** 382
- STORAGE_DIRECTORY parameter**
 - user tracking log files, selecting an output location for 249
- string functions**
 - about 388
 - Abs, about calculating the absolute value 394
 - Acos, calculates the arc cosine of a numerical expression 394
 - ASCII, about converter single character string to 388
 - Asin, about calculating the arc sine of a numerical expression 395
 - Atan, about calculating the arc tangent of a numerical expression 395
 - Atan2, about calculating the arc tangent of y/x 395
 - Bit_Length, about returning length in bits 388
 - Ceiling, about rounding a noninteger numerical expression 395
 - Char, about converting a numerical value 388
 - Char_Length, about returning length in number of characters 389
 - Concat, about forms of function 389
 - Cos, about calculating the cosine of a numerical expression 396
 - Cot, about calculating the cotangent of a numerical expression 396
 - Degrees, about converting an expression from radians to degrees 396
 - Exp, about sending the value e to the power

specified 396

Floor, about rounding a noninteger numerical expression 397

Insert, about inserting a character string 389

Left, about returning characters from the left of a string 390

Length, about returning the length in number of characters 390

Locate, about returning the numerical position of the character_expression1 390

LocateN, about returning the numerical position of the character_expression1 391

Log, calculated the natural logarithm of an expression 397

Log10, about calculating the base 10 logarithm of an expression 397

Lower, about converting a character string to lower case 391

Mod, about dividing the first numerical expression 397

Octet_Length, about returning the bits in base 8 units 391

Pi, about returning the value of pi 397

Position, about returning the numerical position of the character_expression1 392

Power, about taking the first numerical expression to the power specified 398

Radians, about converting from degrees to radians 398

Rand, about returning a pseudo-random number 398

RandFromSeed, about returning a pseudo-random number from a seed value 398

Repeat, returns a specified expression n times 392

Replace, about replacing specified characters 392

Right, about returning a specified number of characters from the right of the string 392

Round, about rounding a numerical expression to n digits 399

Sign, about returning a value of 1, -1, or 0 399

Sin, calculated the sine of a numerical expression 399

Sqrt, about calculating the square root of the numerical expression argument 399

Substring, about creating a new string starting from a fixed number 393

Tan, about calculates the tangent of a numerical expression 400

TrimBoth, about stripping specified leading and trailing characters 393

TrimLeading, about stripping specified leading characters 393

TrimTrailing, about stripping specified trailing characters 394

Truncate, about truncating a decimal number 400

Upper, about converting a character string to uppercase 394

Substring string function, about 393

Sum aggregate function, about 382

SumDistinct aggregate function, about 383

symbols, described (table) 25

Synchronize Aliases utility, about and procedure 214

system

session variables, about and LDAP authentication 337

SQL functions, about 410

variables, about and external table authentication 338

System subject area

about 166

T

table sources

content definitions, creating 116

tables

event polling table, identifying 215

one-to-many relationship, about and diagram 50

tables, sorting in alphanumeric order 139

Tan math function, about 400

TCP/IP, client/server communication method changed 280

text strings, using the Externalize Strings utility to translate 215

Times Series Wizard, about and starting 213

TimestampAdd calendar date/time function, about 404

TimeStampDiff calendar date/time function, about 406

toolbar

docking 24

on/off, toggling 24

Tools menu, described 23

TopN aggregate function, about 383

TrimBoth string function, about 393

TrimLeading string function, about 393

TrimTrailing string function, about 394

troubleshooting

- Analytics Server fails to start 240
- event polling table 272

Truncate math function, about 400

Turn Off Compare Mode, enabling 153

U**UNIX**

- shutting down the Analytics Server 241
- starting the Analytics Server 238

Update Physical Layer wizard, about and starting 216

Upper string function, about 394

usage tracking log files

- sample scripts, locating and example 413
- usage tracking data (table) 414

usage tracking, administering

See also Session Manager; usage tracking log files

- about and example 246
- file naming conventions, about and example 250
- note, about error accessing usage tracking output file 249
- output file, about and schema described (table) 250
- output file, column behavior in 252
- output file, format of 250
- output location, selecting 249
- performance considerations, about and checkpoints described 252

user community, about publishing to 146

user ID, changing for the Analytics Server 239

user interface components

- Edit menu, described 22
- File menu, described 22
- Help menu, described 23
- icons and symbols (table) 25
- keyboard shortcuts (table) 24
- main window, described 21
- Manage menu, described 23
- toolbar functions, described 24
- Tools menu, described 23
- View menu, described 22
- Window menu, described 23

User system function, about 410

USER system session variable, about 300

USER_LOG_FILE_SIZE parameter, using to control query log file size 243

users

- See also* privileges
- Administration account, about and

password 329

- LDAP, using to import users 333
- new user, adding to repository 328
- user accounts, about 328

utilities

- Analytics Event Tables utility, about and starting 215
- copy business model with presentation catalog utility, about and procedure 104
- Externalize Strings utility, about and starting 215
- log viewer utility, interpreting the log records 245
- log viewer utility, using 245
- Repository Documentation utility, about and starting 217
- Synchronize Aliases, about and procedure 214

V

VALUEOF() conversion function, about 409

ValueOf() expressions, using 359

Variables, Manage menu option, described 23

variables, using

- See also* Initialization Block dialog box; initialization blocks
- Analytics Server, re-initializing when it starts 311
- Default initializer column, about value 311
- dynamic repository variables, about and example 298
- new variables, creating 302
- nonsystem session variables, about using 301
- session variables, about 299
- static repository variables, about and example 297
- static repository variables, using variables in expression builders 298
- system session variables, about and LDAP authentication 337
- system session variables, about using 300
- system session variables, table of 300
- system variables, about and external table authentication 338
- Variable Manager, about and classes of variable 297

View menu, described 22

virtual physical tables, creating using the Object Type option 81

W

WEBGROUPS system session variable,
about 301

Week_Of_Quarter calendar date/time
function, about 407

Week_Of_Year calendar date/time function,
about 407

WHERE clause

filter, about and example 119
syntax, about 369

Window menu, described 23

Windows

Analytics Server, configuring for automatic
startup 238
Cluster Controller and Analytics Server,
starting manually 316
NT and 2000 Administrator account, about
and Analytics Server Administrator
account 329
NT and 2000, user ID to access nonlocal
files 348
ODBC data source names, configuring 279
shutting down Analytics Server from a
command prompt 241
shutting down Analytics Server from
Services 240
starting Analytics Server from Services 237

wizards

Calculation wizard, about 33
Rename Wizard, about and starting 216
Replace Wizard, about and starting 214
Repository Import Wizard, about and
using 146
Time Series Wizard, about and starting 213
Update Physical Layer Wizard, about and
starting 216

workspace, Administration Tool

Presentation Catalog dialog box, about using
Presentation Table tab 137
Presentation Tables dialog box, about using
Columns tab 138

X**XML** data source

Ask DBMS, about availability of 65

connection pool properties, setting 78
physical table, setting properties for 92
properties, about setting 77
query output format settings, specifying 79
refresh interval, setting 276

XML Gateway

See also Data Mining Adapter; XML ODBC
database type; XML Gateway
about using 348
example 350
example, more complex 352
HTML tables, accessing 356
HTML tables, example 356
XML data, importing using the XML
Gateway 349
XML examples 362

XML ODBC database type

See also Data Mining Adapter; XML ODBC
database type; XML Gateway
example 361
importing XML data 361
XML data sources, about accessing 361
XML examples 362

XML, using as a data source

See also Data Mining Adapter; XML ODBC
database type
about 347
example, more complex 352
HTML tables, accessing 356
HTML tables, example 356
importing data using the XML Gateway 349
XML examples 362
XML Gateway example 350
XML Gateway, about using 348
XML URL, locating 347
XPath expressions, support of 348
XSL transformation files (XSLT), support
of 348

XMLA

multidimensional data source, importing 61
physical layer, creating 60

Y

Year calendar date/time function,
about 408

