



SIEBEL 7
eBusiness

SIEBEL ANALYTICS SERVER ADMINISTRATION GUIDE

VERSION 7.7
DECEMBER 2003

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404
Copyright © 2003 Siebel Systems, Inc.
All rights reserved.
Printed in the United States of America

No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photographic, magnetic, or other record, without the prior agreement and written permission of Siebel Systems, Inc.

Siebel, the Siebel logo, TrickleSync, TSQ, Universal Agent, and other Siebel product names referenced herein are trademarks of Siebel Systems, Inc., and may be registered in certain jurisdictions.

Other product names, designations, logos, and symbols may be trademarks or registered trademarks of their respective owners.

U.S. GOVERNMENT RESTRICTED RIGHTS. Programs, Ancillary Programs and Documentation, delivered subject to the Department of Defense Federal Acquisition Regulation Supplement, are “commercial computer software” as set forth in DFARS 227.7202, Commercial Computer Software and Commercial Computer Software Documentation, and as such, any use, duplication and disclosure of the Programs, Ancillary Programs and Documentation shall be subject to the restrictions contained in the applicable Siebel license agreement. All other use, duplication and disclosure of the Programs, Ancillary Programs and Documentation by the U.S. Government shall be subject to the applicable Siebel license agreement and the restrictions contained in subsection (c) of FAR 52.227-19, Commercial Computer Software - Restricted Rights (June 1987), or FAR 52.227-14, Rights in Data—General, including Alternate III (June 1987), as applicable. Contractor/licensor is Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404.

Proprietary Information

Siebel Systems, Inc. considers information included in this documentation and in Siebel eBusiness Applications Online Help to be Confidential Information. Your access to and use of this Confidential Information are subject to the terms and conditions of: (1) the applicable Siebel Systems software license agreement, which has been executed and with which you agree to comply; and (2) the proprietary and restricted rights notices included in this documentation.

Contents

Introduction to Siebel Analytics Server Administration Guide

What's New in Siebel Analytics Server Administration Guide, Version 7.7	16
Related Documentation	17

Chapter 1. Overview of the Siebel Analytics Server

The Siebel Analytics Server and Business Modeling	20
Siebel Analytics Server Features	22
Siebel Analytics Server in the Decision Support Environment	24
Siebel Analytics Server and the Data Warehouse	27
Siebel Analytics Server Components	30

Chapter 2. Data Modeling

Understanding the Business Model	35
Understanding the Physical Database Model	37
Primary Key-Foreign Key Relationships	42
Knowing the Contents of the Physical Database	43
Knowing the Aggregate Table Definitions	43
Logical Business Models	44
Understanding Dimensional Models	44

Chapter 3. Administration Tool Basics

Analytics Administration Tool User Interface Components	51
Online and Offline Repository Modes	60

Checking the Consistency of a Repository or a Business Model	63
Checking In Changes	64
Setting Preferences	65
Using the Options Dialog Box—General Tab	65
Using the Options Dialog Box—Repository Tab	67
Using the Options Dialog Box—Sort Objects Tab	68
Using the Options Dialog Box—Cache Manager Tab	69
Using the Options Dialog Box—More Tab	69
Setting Permissions for Repository Objects	70
Editing, Deleting, and Reordering Objects in the Repository	70
Displaying and Updating Row Counts for Tables and Columns	71
Using the Browse Dialog Box	72

Chapter 4. Setting Up a Siebel Analytics Server Repository

About Repository Structure in the Administration Tool	75
Process of Setting Up a Repository	78
Creating a New Analytics Repository File	78

Chapter 5. Creating and Administering the Physical Layer in a Repository

Process of Creating the Physical Layer by Importing a Physical Schema	82
Importing a Physical Schema	83
Process of Creating the Physical Layer for a Multidimensional Data Source	87
Setting Up Database Objects	88
Creating a Database Object in the Physical Layer	88
Specifying SQL Features Supported by a Database	90
Setting up Connection Pools	91
Creating a Connection Pool for All Data Sources	93
Setting Up Additional Connection Pool Properties for an XML Data Source	97
Setting Additional Connection Pool Properties for a Multidimensional Data	

Source 99

About Physical Tables 100

Creating and Administering Physical Tables 102

 Creating and Administering General Properties for a Physical Table 103

 Creating and Administering Columns and Keys in a Physical Table 105

 Adding and Administering Hierarchies in the Physical Layer for a
Multidimensional Data Source 109

 Setting Physical Table Properties for an XML Data Source 114

Creating Physical Layer Folders 114

 Creating Catalog Folders 114

 Creating Schema Folders 115

 Using a Variable to Specify the Name of a Catalog or Schema 115

 Setting Up Display Folders in the Physical Layer 116

About Physical Joins 117

Defining Physical Foreign Keys and Joins 120

 Defining Physical Foreign Keys or Complex Joins with the Joins Manager . . 120

 Defining Physical Joins in the Physical Diagram 121

Using Database Hints 123

**Chapter 6. Creating and Administering the Business Model
and Mapping Layer in a Repository**

Creating Business Model Objects 128

Creating and Administering Logical Tables 129

 Creating Logical Tables 129

 Adding or Editing Logical Table Sources 131

 Specifying a Primary Key in a Logical Table 133

 Editing Foreign Keys for a Logical Table 134

Creating and Administering Logical Columns 134

 Creating a Logical Column 135

 Setting Default Levels of Aggregation for Measure Columns 136

Associating an Attribute with a Level in Dimension Tables	136
Creating and Administering Logical Table Sources (Mappings)	137
Creating or Removing a Logical Table Source	138
Defining Physical to Logical Table Mappings	139
Defining Content of Sources	141
About Dimensions and Hierarchical Levels	143
Process of Creating and Administering Dimensions	143
Creating Dimensions	144
Creating Dimension Levels and Keys	144
Creating Dimensions Automatically	151
Setting Up Dimension-Specific Aggregate Rules for Logical Columns	153
Setting Up Display Folders in the Business Model and Mapping Layer	156
Defining Logical Joins	157
Defining Logical Joins with the Joins Manager	157
Defining Logical Joins with the Business Model Diagram	160
Specifying a Driving Table	162
Identifying Physical Tables That Map to Logical Objects	163

Chapter 7. Creating and Maintaining the Presentation Layer in a Repository

Creating the Presentation Layer in the Repository	165
Presentation Layer Objects	167
Working with Presentation Catalogs	168
Working with Presentation Tables	171
Working with Presentation Columns	172
Using the Alias Tab of Presentation Layer Dialog Boxes	174

Chapter 8. Completing Setup and Managing Repository Files

Process of Completing the Setup for a Repository File	175
Saving the Repository and Checking Consistency	176

Add an Entry in the NQSConfig.INI File	177
Create the Data Source	177
Start the Siebel Analytics Server	178
Test and Refine the Repository	178
Publish to User Community	178
Importing from Another Repository	179
Querying and Managing Repository Metadata	180
Constructing a Filter for Query Results	184
Comparing Repositories	186
Merging Siebel Analytics Repositories	188
Creating an Analytics Multiuser Development Environment	191
Business Scenario for Analytics Multiuser Development	191
Setting up an Analytics Multiuser Development Environment	192
Making Changes in an Analytics Multiuser Development Environment	194
Setting Up a Developer’s Machine for Analytics Multiuser Development	195
Changing Metadata in an Analytics Multiuser Development Environment	195
Merging Metadata Changes Into the Source Repository	198

Chapter 9. Setting Up Disconnected Analytics

Disconnected Analytics Architecture (Server and Laptop)	203
Installing Siebel Disconnected Analytics	205
Process of Creating and Deploying Disconnected Analytics Applications	206
Creating and Testing the SQL Anywhere Database Tables	206
Design the Table Schema for the Disconnected Analytics Application	207
Write the DDLs to Create the Local Database Tables	207
Creating and Testing Local Database Tables	209
Creating the Disconnected Analytics Repository	210
Create the Physical Layer of the Disconnected Analytics Repository	211
Create the Business Model and Mapping Layer of the Disconnected Repository	213

Create the Presentation Layer of the Disconnected Repository	214
Testing the Disconnected Repository	214
Creating the Disconnected Web Catalog	215
Defining Sourcing Reports for Disconnected Analytics	216
Defining Data Sets for Disconnected Analytics	218
Creating the Disconnected Application Configuration File	219
Creating the Application Configuration File for Disconnected Analytics	220
Preparing Disconnected Analytics Applications for Deployment	226
Testing and Deploying Siebel Disconnected Analytics	229
Testing Disconnected Analytics Applications	229
Testing the Installation Processes	230
Installing an Application Using Disconnected Analytics Application Manager	230
Installing a Disconnected Analytics Application Silently	231
Deleting Disconnected Analytics Applications	232

Chapter 10. Administration Tool Utilities and Expression Builder

Utilities and Wizards	233
Time Series Wizard	234
Synchronize Aliases	234
Replace Wizard	235
Copy Business Model with Presentation Catalog	236
Siebel Analytics Event Tables	236
Execute UDML	237
Externalize Strings	237
Rename Wizard	238
Update Physical Layer Wizard	239
Generating Documentation of Repository Mappings	241
Expression Builder	241

Chapter 11. Setting Up Aggregate Navigation

Specify the Aggregate Levels for Each Source	250
Create Dimension Sources for Each Level of Aggregated Fact Data	250
Specify Fragmentation Content	253
Aggregate Table Fragments	259

Chapter 12. Administering the Query Environment

Starting the Siebel Analytics Server	265
Starting the Server from the Services Applet in Windows	265
Configuring the Server for Automatic Startup in Windows	266
Running the Siebel Startup Script in UNIX	267
Changing the User ID in Which the Siebel Analytics Server Runs	267
If the Server Fails to Start	268
Shutting Down the Server	268
Shutting Down the Server from the Services Applet in Windows	269
Shutting Down the Server from a Command Prompt in Windows	270
Running the Siebel Analytics Server Shutdown Script in UNIX	271
Shutting Down the Server Using the Administration Tool	271
Getting Users to Connect to the Server	272
Administering the Query Log	272
Administering Usage Tracking	277
Administering Direct Insertion for Usage Tracking	277
Selecting an Output Location	279
Performance Considerations	283
Collecting More Detailed Information About Queries	283
Server Session Management	284
Server Configuration and Tuning	287

Chapter 13. Query Caching in Siebel Analytics Server

About the Analytics Server Query Cache	289
--	-----

Query Cache Architecture	292
Configuring Query Caching	292
Monitoring and Managing the Cache	295
Purging Cache Programmatically	297
Strategies for Using the Cache	298
Cache Event Processing with an Event Polling Table	302
Setting Up Event Polling Tables on the Physical Databases	304
Making the Event Polling Table Active	307
Populating the Siebel Analytics Server Event Polling Table	309
Troubleshooting Problems with an Event Polling Table	309
Making Changes to a Repository	310
Using the Cache Manager	311
Displaying Global Cache Information	312
Purging Cache	313
About the Refresh Interval for XML Data Sources	315

Chapter 14. Connectivity and Third-Party Tools

Configuring Siebel Analytics ODBC Data Source Names (DSNs)	317
Third-Party Tools and Relational Datasource Adapters	320
Importing Metadata	320
ODBC Conformance Level	321

Chapter 15. Using Variables in the Analytics Server Repository

Using the Analytics Variable Manager	325
Using Repository Variables	326
About Session Variables	328
Creating New Variables	331
About Initialization Blocks	333
Initializing Dynamic Repository Variables	333
Initializing Session Variables	334

Creating and Editing Initialization Blocks 336
 Tasks Using the Initialization Block Dialog Box—Variable Tab 341
 Execution Precedence 343

Chapter 16. Clustering Siebel Analytics Servers

About the Cluster Server Feature 345
 Components of the Cluster Server Feature 345
 Implementing the Cluster Server Feature 347
 Chronology of a Cluster Operation 351
 Using the Cluster Manager 353
 Viewing Cluster Information 354
 Managing Clustered Servers 359
 Performance Considerations 360

Chapter 17. Security in Siebel Analytics

Analytics Security Manager 361
 Working with Users 362
 Working with Groups 365
 Importing Users and Groups from LDAP 371
 Authentication Options 374
 Operating System Authentication 375
 LDAP Authentication 376
 External Table Authentication 378
 Database Authentication 381
 Siebel Analytics Server Internal Authentication 382
 Order of Authentication 383
 Bypassing Siebel Analytics Server Security 383
 Managing Query Execution Privileges 384

Chapter 18. Using XML as a Data Source for Analytics

Locating the XML URL	389
Using the Siebel Analytics Server XML Gateway	391
Siebel Analytics Server XML Gateway Example	394
Accessing HTML Tables	402
Using the Data Mining Adapter	404
Using XML ODBC	409
XML ODBC Example	410
XML Examples	411
83.xml	412
8_sch.xml	413
84.xml	414
Island2.htm	415

Chapter 19. SQL Reference

SQL Syntax and Semantics	417
SELECT Query Specification Syntax	418
SELECT Usage Notes	419
SELECT List Syntax	420
Rules for Queries with Aggregate Functions	421
SQL Logical Operators	427
Conditional Expressions	427
SQL Reference	429
Aggregate Functions	430
Running Aggregate Functions	439
String Functions	445
Math Functions	453
Calendar Date/Time Functions	460
Conversion Functions	471
System Functions	474
Expressing Literals	474

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

Create Table Scripts for Usage Tracking Data 477
 Loading Usage Tracking Tables with Log Files 478
 Description of the Usage Tracking Data 478

Appendix B. Pharma Analytics Administration Reference

Sourcing Reports for Pharma Analytics 482
 Data Sets for Pharma Analytics Data 483

Index

Introduction to Siebel Analytics Server Administration Guide

Siebel Analytics Server Administration Guide provides information to create and maintain Siebel Analytics repositories and to administer the Siebel Analytics Server environment. This document includes conceptual and step-by-step information.

Intended users of this guide include system administrators for the Siebel Analytics environment (referred to as Siebel administrators), architects of the decision support applications created with Siebel Analytics software, and managers who are responsible for the installation, development, and system administration of decision support applications.

This book will be useful primarily to people whose titles or job descriptions match one of the following:

Database Administrators	Persons who administer the database system, including data loading, system monitoring, backup and recovery, space allocation and sizing, and user account management.
Siebel Application Administrators	Persons responsible for planning, setting up, and maintaining Siebel applications.
Siebel Application Developers	Persons who plan, implement, and configure Siebel applications, possibly adding new functionality.
Siebel System Administrators	Persons responsible for the whole system, including installing, maintaining, and upgrading Siebel applications.

This guide assumes that you are knowledgeable in the areas of relational databases, decision support systems, dimensional design, and the operating system under which you are running the Siebel Analytics components.

What's New in Siebel Analytics Server Administration Guide, Version 7.7

Table 1 lists changes described in this version of the documentation to support version 7.7 of the Siebel Analytics software.

NOTE: Siebel Analytics version 7.7 is compatible with Siebel eBusiness Applications version 7.5.3.

Table 1. New Product Features in Siebel Analytics Server Administration Guide, Version 7.7

Topic	Description
“Setting Up Disconnected Analytics” on page 201	Setup instructions for Siebel Disconnected Analytics.
“Creating an Analytics Multiuser Development Environment” on page 191 and “Making Changes in an Analytics Multiuser Development Environment” on page 194	Administrator setup instructions and instructions about how to use a multiuser development environment.
“Process of Creating the Physical Layer for a Multidimensional Data Source” on page 87	Information about how to create the physical layer for a multidimensional data source.
“Importing a Physical Schema” on page 83	New features that apply to importing objects into the Physical layer are in the following list: <ul style="list-style-type: none">■ New option to Import Through Server on the File menu.■ Can create a table mask for importing.■ You can import using a text (CSV or TXT) file
“Setting Up Dimension-Specific Aggregate Rules for Logical Columns” on page 153	Instructions that allow you to set aggregation rules for multiple logical columns.

Table 1. New Product Features in Siebel Analytics Server Administration Guide, Version 7.7

Topic	Description
“Creating Dimensions Automatically” on page 151	Instructions for setting up a dimension automatically from a logical dimension table.
“Icons and Symbols in the Administration Tool” on page 56	Added new icons.
“Associating a Logical Column and Its Table With a Dimension Level” on page 148	Instructions for showing the association between a dimension and an logical table.
“Externalize Strings” on page 237, “Generating Documentation of Repository Mappings” on page 241, and “Querying and Managing Repository Metadata” on page 180.	Unicode and UTF-8 are new save formats in the Externalize Strings utility, Repository Documentation utility, and Query Repository dialog box.
Appendix A, “Usage Tracking Data Descriptions and Using the Log File Method”	Instructions for managing usage tracking.
Appendix B, “Pharma Analytics Administration Reference”	Information about Disconnected Analytics sourcing reports and data sets tailored for Pharma Analytics.

Related Documentation

For more information about Siebel Analytics, read the following books.

- *Siebel Analytics Installation and Configuration Guide*
- *Siebel Disconnected Analytics Online Help*
- *Analytics Web Online Help*

Overview of the Siebel Analytics Server

1

This section briefly describes how the Siebel Analytics Server can help an organization meet its decision support goals. It also provides an overview of the architecture of the Analytics server and describes some of the tasks to consider when constructing a data warehouse.

The Siebel Analytics Server and Business Modeling

The Siebel Analytics Server is the core server behind Siebel Analytics. Siebel Analytics provides the power behind Siebel Intelligence Dashboards for access and analysis of structured data that is distributed across an organization or an the supply chain for an organization. The Analytics server allows a single information request to query multiple data sources, providing information access to members of the enterprise and, in Web-based applications, to suppliers, customers, prospects, or any authorized user with Web access.

Business organizations often find that the success of a strategic or tactical initiative depends on correctly monitoring, predicting, and understanding business events. Organizations also find that the use of technology to provide information to support decisions is critical. Providing that information with the appropriate technology remains, however, a significant challenge.

Given this complexity, analysis comes to a standstill—or worse, proceeds with inaccurate or misinterpreted results. In cases where attempts are made to solve this problem, the result is often a collection of stand-alone data marts designed to answer the current needs of a particular user group. Subsequent user groups trying to answer business questions might not know of the existence of these data marts, might find more than one data mart that partially answers their questions, or might find more than one that answers the questions with different results.

Business Model to SQL

The Siebel Analytics Server presents a business model that is accessible through structured query language (SQL). SQL is the standard interface to enterprise relational database management systems (RDBMS). Because the business model presented can be a simple one, you can query these models with simple SQL.

The Siebel Analytics Server provides a bridge from the simple SQL needed to query the business model to the often complicated SQL needed to extract the requested information from the underlying databases. It builds this bridge through the following methods:

- Providing a business model that is accessible through SQL
- Mapping the business model to the underlying data sources

Business models are dimensional models. Dimensions describe the attributes and hierarchies of the business. The models also include business measures useful for monitoring or predicting business performance. Because they mirror the way people think about business information, dimensional models can be understood by nontechnical business analysts.

The SQL required to query a Siebel Analytics Server business model is even less complex than the SQL required to query an RDBMS dimensional model. Join information, aggregation functions, and Group By clauses are not needed when querying through the Analytics server. Its server-side capabilities, together with its data modeling capabilities, provide that bridge, which allows simple SQL to pose complex business questions, regardless of where the data is physically located and independent of the physical model in which the data is stored.

Siebel Analytics Server Features

The Siebel Analytics Server is designed to address the issues that arise in a data warehouse environment. A data warehouse environment includes all of the components to build decision support applications—databases, client tools, data extraction and movement tools, detailed procedures, and planning.

Multi-Database Access

Data typically exists in many different places. Instead of moving data to a single database, the Siebel Analytics Server can access multiple databases simultaneously to answer a single query, regardless of whether the databases are from different vendors or what version of database software each runs under. This heterogeneous environment allows the Siebel Analytics Server administrator to set up decision support systems that would traditionally require a great deal of data extraction effort to implement. When the Analytics server accesses heterogeneous databases, any required transformation is handled automatically, such as datatype casting or applying multiple complex functions. The Analytics server can also access XML data sources.

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

Aggregate Navigation

Data warehouses often include precomputed aggregates to improve query performance. However, adding aggregate tables increases the complexity of the query environment, because users need to know that they should query the aggregate tables directly. They need to analyze the meaning of a query and pick the appropriate aggregate tables, which requires them to understand how the tables are set up. The Siebel Analytics Server allows users to see a single view of the data, and if they ask for information that is precomputed in aggregate tables, the server automatically uses those aggregates to retrieve the data.

Query Caching

Aggregate tables and aggregate navigation help speed up warehouse queries that someone thought to precompute. Query caching stores results for queries that users actually run. The Siebel Analytics Server administrator can configure the server to cache query results for use by subsequent queries. The cached results act much like precomputed aggregates in that they can dramatically accelerate query performance. All security attributes remain in force for the cached results, so users can access only the data they are authorized to view. Queries can be scheduled and the results cached to make information instantly accessible. To ensure the information stored in the cache is current with the underlying databases, the Siebel Analytics Server administrator can set up cache purging rules.

Query Speed

Query speed is always an issue. The Siebel Analytics Server features such as query caching, aggregate table navigation, database-specific SQL optimization, connection pooling, parallel query processing, high-performance sorting, and efficient multi-table join technology provide faster query response times, sometimes enabling online analysis where it was not possible before.

Metadata Repositories

Data warehouses typically have many components, each having its own security attributes and complexities. This can add expensive administrative overhead to the data warehouse environment. With the Siebel Analytics Server, all the rules needed for security, data modeling, aggregate navigation, caching, and connectivity is stored in metadata repositories. Each metadata repository can store multiple business models. The Siebel Analytics Server can access multiple repositories.

Siebel Analytics Server in the Decision Support Environment

The Siebel Analytics Server serves as a portal to structured data that resides in one or more data sources—multiple data marts, the Siebel Data Warehouse, an enterprise data warehouse, an operational data store, transaction system databases, personal databases, and more. Transparent to both end users and query tools, the Analytics server functions as the integrating component of a complex decision support system (DSS) by acting as a layer of abstraction and unification over the underlying databases. This offers users a simplified query environment in which they can ask business questions that span information sources across the enterprise and beyond. This section shows how the Analytics server fits into the decision support environment.

Components of the Environment

Figure 1 shows the Siebel Analytics Server along with other components of the data warehouse.

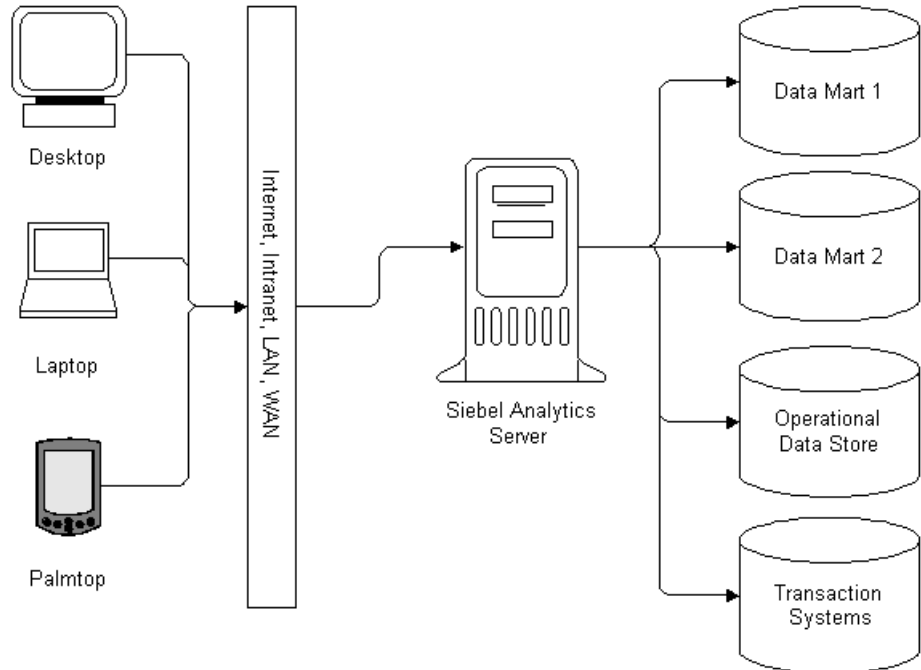


Figure 1. High-Level Environment Diagram

The Siebel Analytics Server is designed to complement decision support systems, not replace them. Well-designed systems allow business users to analyze large quantities of historic information in several forms—detail, snapshot, and summary. Each type of information has a specific use:

- Detail data provides insights into the specifics of the business. For example, such analyses can lead businesses to maximize product strategies or cut marketing costs.

Overview of the Siebel Analytics Server

Siebel Analytics Server in the Decision Support Environment

- Snapshot data shows a view of the business data at a given time. Snapshot data can be used to understand trends and to monitor business performance. Product profitability, customer segmentation, and shelf management strategies all benefit from snapshot data.
- Summary data can be used to generate reports at a higher level of aggregation to monitor performance at the upper levels of the business organization. Analysts needing more detailed information can drill down to lower-level snapshot or detail data.

Siebel Analytics Server and the Data Warehouse

The data warehouse approach requires careful planning to succeed in the goal of providing ready access to data. Constructing a new data warehouse typically involves these tasks:

- Defining a business model for the data
- Retrieving source data from transaction systems, and then cleansing, transforming, and aggregating the data so it can be loaded into queryable systems
- Selecting software and building custom applications
- Integrating various software components and making multiple data sources available to users
- Defining users and setting up the proper level of security access
- Publishing the data warehouse to the user community

Siebel Analytics Server functionality addresses these tasks and makes them easier to accomplish.

Siebel Analytics Server and Business Model Prototyping

One challenge is to present a user-friendly, straightforward view of business information to end users. Regardless of the physical characteristics of the source data, the business model needs to be developed in conjunction with the people who know and analyze the business, using their terminology and nomenclature. The Siebel Analytics Server allows you to develop and test working prototypes that deliver fully functioning business models with security. Based on feedback, this process can iterate repeatedly before a business undertakes expensive processes to move and reorganize data at the physical level.

Siebel Analytics Server and Source Data

The Siebel Analytics Server provides some kinds of data transformations at query time while leaving source data in place. This allows business model prototyping and rapid decision support system development. For example, the data used in data warehouses typically comes from multiple sources, including diverse transaction systems, several departments, or independent third-party vendors. Normally, various data transformations need to take place to change the format of the data, rename it, perform operations such as summing or averaging, convert it from one format to another, multiply it by equivalency factors, and make data from one system compatible with another.

The Siebel Analytics Server can also help identify potentially erroneous data sources or relationally invalid data sources (dirty data) and, if rules for filtering the data are understood, prevent dirty data from reaching users.

Siebel Analytics Server and Availability for Applications

Because the Siebel Analytics Server does not have a proprietary interface, data is available to any application using standard SQL queries and ODBC (or JDBC to ODBC bridge) connectivity, including both Web-based and client/server applications.

Siebel Analytics Server and OLAP Tools

Normally, the choice of an online analytical processing (OLAP) tool dictates the structure of the data in the data warehouse. The Siebel Analytics Server eliminates this restriction because it can present multiple business models for the same physical data sources. This allows you to customize the presentation of data to your specific online analytical processing (OLAP) tools.

Siebel Analytics Server and Security

The Siebel Analytics Server provides secure access control for both databases and users. The Siebel Analytics Server administrator can grant rights to users through groups, by explicit configuration, or a combination of the two. The Analytics server supports Windows NT and Windows 2000 unified logon, and can integrate with external security systems, such as Lightweight Directory Access Protocol (LDAP) or external database tables that are already in place. Data can be filtered so that users see only the data rows that are pertinent to their role in the organization.

Siebel Analytics Server and Access for End Users

Publishing easy-to-use, realistic business models is at the core of any successful query environment. After you have used the Siebel Analytics Server to create, tailor, and test user-friendly business models, they can be published to your user community. If you want to use the Web to publish, publishing consists of providing the appropriate URL and a logon and password to your users.

Unified Query Environment

In many data warehouse environments, users have to learn different tools to query different parts of the business. One of the goals of a data warehouse is to make all information accessible to the users from a single environment. The Siebel Analytics Server acts as a layer of abstraction over the underlying databases, and offers users a unified query environment in which they can ask business questions that span information sources across the enterprise.

Security is controlled directly through the Analytics server, so access can be granted as the security policy dictates. Multiple business models can be set up as multiple data sources to the users. The Siebel Analytics Server administrator can design the environment based on user wants, needs, and the security policies of the organization. Security can be regulated for each user or group of users and privileges can be set to restrict access to any piece of information, no matter how fine-grained.

Siebel Analytics Server Components

[Figure 2 on page 31](#) shows the architectural components of the Siebel Analytics Server and how they fit into the enterprise.

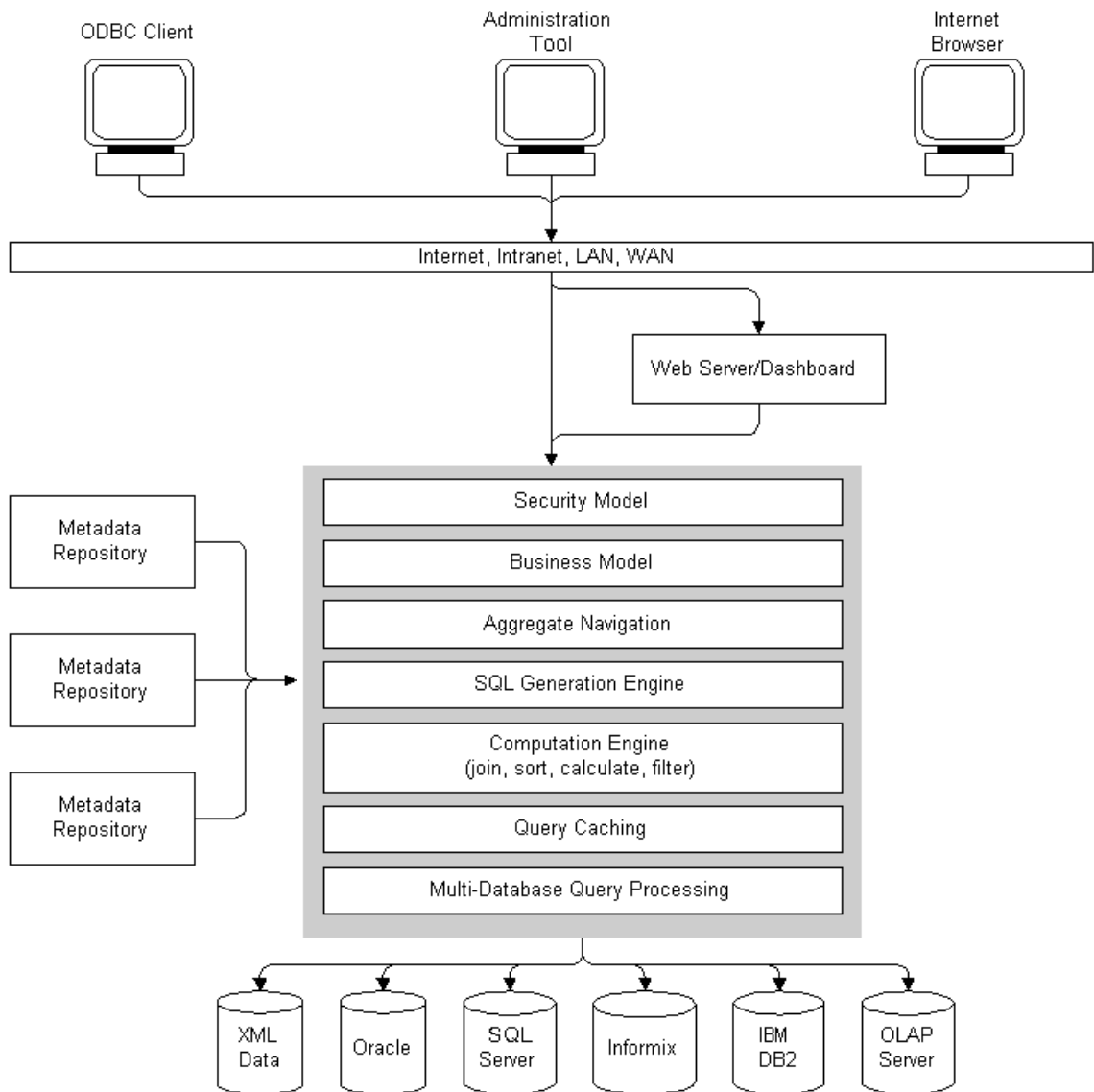


Figure 2. Architectural Components

Multithreaded Architecture

In Windows, the Siebel Analytics Server runs as a multithreaded service, and in UNIX, as a multithreaded process. The server is the engine behind Siebel Analytics and performs most of the work. Centralizing the work in the server allows thin clients, such as Web browsers, to run queries without any additional client software.

The Siebel Analytics Server multithreaded architecture can scale to multiple processors and thousands of users. The server loads all the metadata stored and processes queries based on what is configured in the repositories. This includes the business model, security access controls, aggregate navigation, and the physical data sources.

The server accepts query requests from Web clients, ODBC clients, and other applications connecting to Siebel Analytics. The server includes database-optimized SQL generation, a full join engine, sophisticated optimization algorithms, advanced analytic calculation capabilities, and an integrated caching mechanism to access and process data in the most efficient way possible.

Metadata Repository

The metadata repositories store all the information about the application environment. All of the security, data modeling, aggregate navigation, caching, and connectivity information is stored in metadata repositories. Each repository can store multiple business models. The Siebel Analytics Server can access multiple repositories.

The Administration Tool

The Administration Tool is a Windows application that allows the Siebel Analytics Server administrator to create and edit repositories. Repositories can be edited in either online or offline mode. While in online mode, the administrator edits a live repository that the server has in memory; that is, users can still access the repository while changes are being made. Changes to the repository are immediately applied to the server when they are checked in. In offline mode, the administrator can edit any repository and save the changes. The changes are applied when the Siebel Analytics Server next starts and loads the repository into memory.

The Administration Tool also includes a session manager and a cache manager which are available in online mode. The Session Manager displays activity. You can see which users are logged in, disconnect users, and terminate queries. The Cache Manager displays what queries have been cached and how frequently they have been accessed. You can also select and delete items in cache.

Web Server and Intelligence Dashboards

The Siebel Analytics Server integrates with Web servers to form information Intelligence Dashboards where users can access predefined requests and view results, as well as perform ad-hoc queries. Siebel Systems Web solutions transform a standard Internet browser into an advanced query tool where results can be displayed and delivered using multiple formats.

An Intelligence Dashboard can be constructed for rapid deployment. This capability can also be used from a third-party packaged portal offering.

Web Clients and Other ODBC Clients

A wide variety of client tools can connect to the Siebel Analytics Server. Using any ODBC-compliant tool, users can perform real-time queries to access enterprise data with the Analytics server. A system might have some users using a tool that also queries existing applications, other users querying using Web browsers, and other users using custom-built applications that access the Analytics server.

NOTE: The client portion of Siebel Analytics includes a tool, Siebel Analytics Client, that allows you to issue SQL to an ODBC data source. This tool is sometimes useful for testing and debugging purposes. However, it does not support double byte character sets. For information about installing the Siebel Analytics Client, see *Siebel Analytics Installation and Configuration Guide*.

Overview of the Siebel Analytics Server

Siebel Analytics Server Components

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.

For both historical and technological reasons, existing physical data structures rarely conform to such business models. Because Siebel Analytics Server repositories include business models and physical data models, the Siebel Analytics Server administrator needs to understand data modeling concepts. This section discusses some of the considerations involved in data models and their relationships to business models.

Understanding the Business Model

Understanding the business model is the first step in developing a usable data model for decision support—a model that business analysts will inherently understand and that will answer meaningful questions correctly. This requires breaking down the 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 the business operates under?
- Are there hierarchical elements in each dimension and what are the parent-child relationships that define each hierarchy?

When you can answer these questions, you can use the Administration Tool to build a valid, usable, and intuitive business model.

Identifying the Factual Measures

Facts are the business measures to be analyzed. These are typically additive items, such as sales dollars and units sold. Each measure will have its own aggregation rule, for example, SUM, AVG, MIN, MAX, or COUNT. Often a business will want to compare values of a measure over time and will need a calculation to express the comparison, as in, for example, a change or percent change.

Identifying the Dimensions of a Business

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. 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 Hierarchical Relationships Between Attributes

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.

Understanding the hierarchies is essential to provide the metadata that allows the Siebel 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.

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.

Understanding the Physical Database Model

The Siebel Analytics Server provides an interface to map the business model to the underlying physical databases. Sometimes you can control the physical design of the underlying databases, and it might be modeled like the business model. But sometimes the database already exists and you have to work with what is there. Regardless of whether you have input into the physical database design, you need to understand both its structure and its content.

NOTE: When creating your repository, you should set up the Physical layer first and then create the Business Model and Mapping layer.

Types of Physical Models

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

Entity-Relationship (E-R) Schemas

The entity-relationship (E-R) model 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 models 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, which contain attributes of the business, and fact tables, which contain individual records with a few facts and foreign keys to each of the dimension tables. Dimensional schemas are very good 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 are excellent for queries that analyze the business across multiple dimensions.

Star Schema

A star schema is a dimensional schema with a single fact table that has foreign key relationships with several dimension tables. [Figure 3](#) shows a sample star schema.

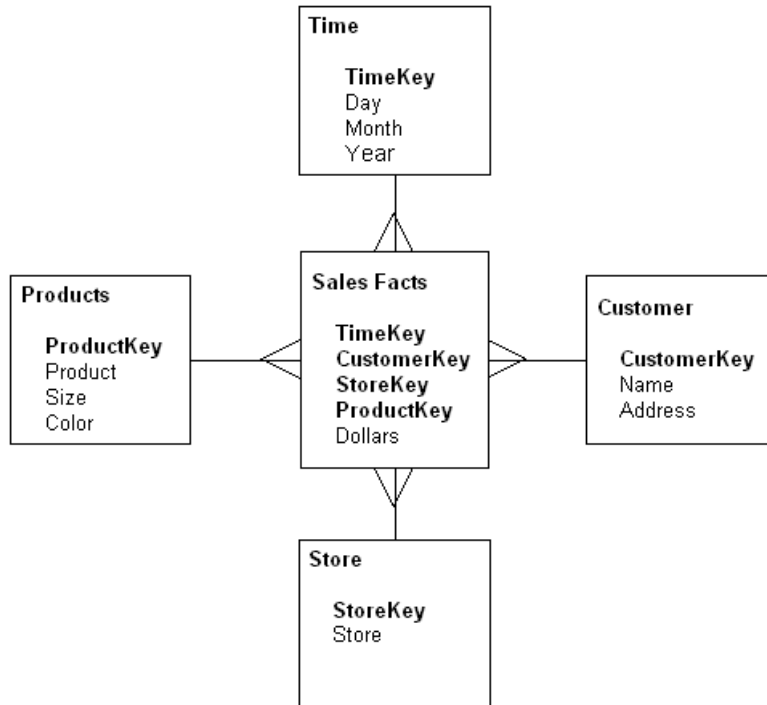


Figure 3. Sample Star Schema

The dimension tables model the business 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 tables, read [“Knowing the Aggregate Table Definitions”](#) on page 43.

Snowflake Schema

A snowflake schema is a dimensional schema where one or more of the dimensions are normalized to some extent. 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. [Figure 4](#) shows a typical snowflake schema where the time dimension has been normalized into a snowflake.

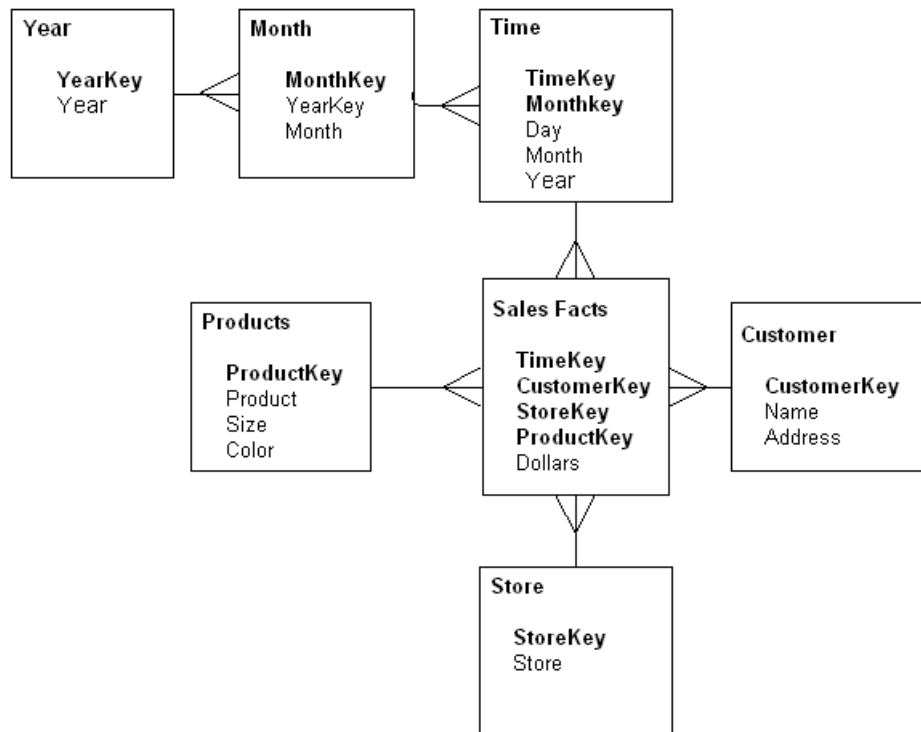


Figure 4. Sample Snowflake Schema

Primary Key-Foreign Key Relationships

To fully understand physical database models, it is important to understand 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 5](#), 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.

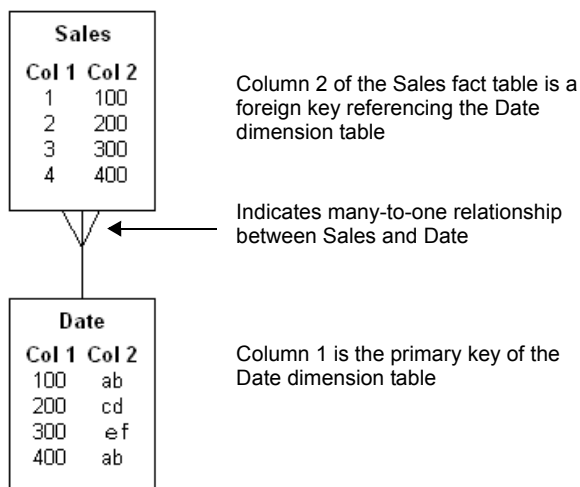


Figure 5. Primary Key-Foreign Key Sample

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.

Knowing the Contents of the Physical Database

The Administration Tool provides an interface to map logical tables to the underlying physical tables in the database. To do this correctly, you need to understand the contents of the physical database. You need to know:

- The contents of each table
- The detail level for each table
- The contents of each column
- How each measure is calculated
- The table definition for each aggregate table
- 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 application developers of the applications that create the data.

Knowing the Aggregate Table Definitions

To set up aggregate navigation, you need to specify the aggregate table definition for each aggregate. Specifically, the information the Siebel Analytics Server requires is:

- 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 Aggregate Navigation” on page 249](#).

Logical Business Models

Using the Administration Tool, the administrator can create one or more logical business models in a repository. Business models are used to define and store business models and to provide the mapping of the business model to the underlying physical databases.

Logical Tables and Columns

Each business model contains two or more logical tables, and each logical table contains one or more logical columns. A logical table is an abstraction above a physical table. It can be a subset of a physical table (a subset of columns or a subset of rows); it can combine the contents of two or more physical tables; or it can be derived from other logical tables. Typically, logical tables reduce complexity in the information model because a single logical table can map to multiple physical tables.

Similarly, a logical column is an abstraction above a physical column. It can be mapped to one or more physical columns, to a scalar expression involving physical columns, or to other logical columns.

Heterogeneous Data Sources

The logical tables and columns can be mapped from multiple data sources. The data sources can originate from multiple databases of the same or of different types. For information about supported databases, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

Understanding Dimensional Models

To analyze business data, the data needs to be mapped logically to a business model. The Siebel 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 Siebel 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.

Dimensional Hierarchies

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 6](#).



Figure 6. 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 Siebel Analytics Server allows you to define the hierarchy definitions for your particular business, however complex, assuring that analyses will conform to your business definitions.

Factual Measures

Factual measures, or facts, are typically additive data such as dollar value or quantity sold, and they can be specified in terms of the dimensions. For example, you might ask for the sum of dollars for a given product in a given market over a given time period. Measures can also be aggregated by applying other aggregation rules, such as averaging instead of summing. Furthermore, the aggregation rules can be specific to particular dimensions. The Siebel Analytics Server allows you to define these complex, dimension-specific aggregation rules.

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.

Bridge Tables to Model Many-to-Many Relationships

Star schemas and snowflake schemas work well for modeling a particular part of a business where there are one-to-many relationships between the dimension tables and the fact tables. However, sometimes it is necessary to model many-to-many relationships between dimension tables and fact 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 table and the dimension table. A bridge table stores multiple records corresponding to that dimension.

To understand how a bridge table works, consider the following portion of a sample health care schema, as shown in [Figure 7](#).

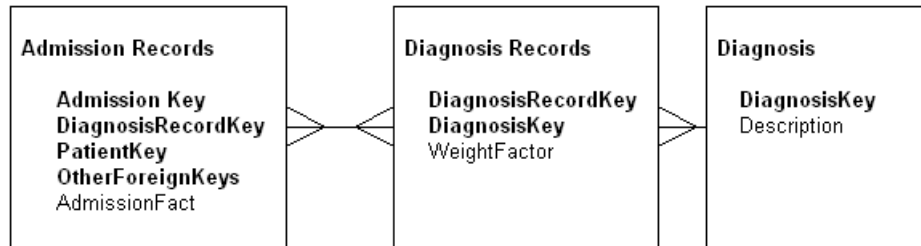


Figure 7. Sample Health Care Schema

The many-to-many relationship is that for each patient admission, there can be multiple diagnoses. For example, a patient can be diagnosed with the flu and with a broken wrist. The bridge table then needs to have a weight factor column in it so that all of the diagnoses for a single admission add up to a value of 1. The weight factor has to be calculated as part of the process of building the data. For the case of the patient diagnosed with the flu and a broken wrist, there would be one record in the Admission Records table, two records in the Diagnosis Record table, and two records in the Diagnosis table, as shown in [Figure 8](#).

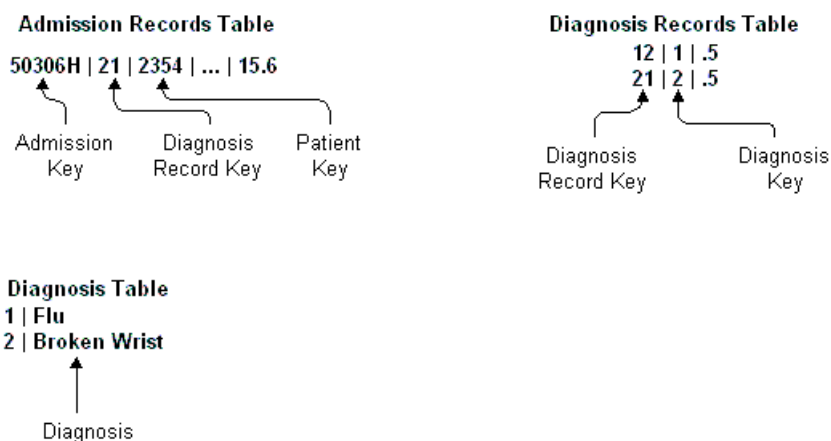


Figure 8. Multiple Diagnoses for One Patient

NOTE: This type of design can create more records in the Diagnosis Records table than in the Admission Records table. You can limit the number of records in the Diagnosis Records table by predefining groups of diagnosis and forcing each admission record to fit in one of these predefined groups.

In the Administration Tool, the Logical Table dialog box has an option you can select to specify that a table is a bridge table.

Single Table Models

For the greatest simplicity for end users, you can 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 Siebel 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 165.

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 Siebel Analytics Server administrator to create and edit repositories.

Analytics Administration Tool User Interface Components

This section includes a description of the following interface components:

- [“Main Window in the Administration Tool”](#)
- [“Menus in the Administration Tool” on page 52](#)
- [“Toolbar in the Administration Tool” on page 54](#)
- [“Keyboard Shortcuts in the Administration Tool” on page 55](#)
- [“Icons and Symbols in the Administration Tool” on page 56](#)
- [“Online Help in the Administration Tool” on page 60](#)

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 Siebel 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. It shows the business models, which 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 9 shows the three layers of a repository, as described in the preceding list.

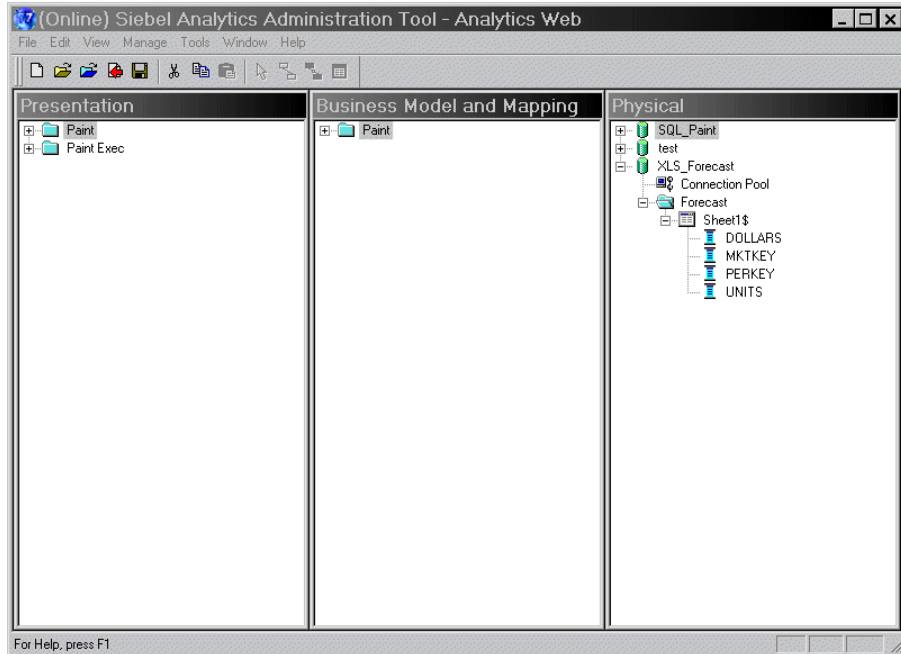


Figure 9. Example Administration Tool Main Window

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 2](#).

Table 2. Manage Menu Functions

Menu Option	Description
Jobs	This option is available when a repository is opened in online mode. It opens the Job Manager window, which is the management interface to Siebel Analytics Scheduler.
Sessions	This option is available when a repository is opened in online mode. It opens the Session Manager, which you can use to 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. It opens the Cache Manager window, which allows you to monitor and manage the cache.
Clusters	This option is available when the Siebel Analytics Cluster Server feature is installed. It opens the Cluster Manager window, which you can use to monitor and manage the operations and activities of the cluster.
Security	This option opens the Security Manager, which displays security information for a repository and provides a central location for security configuration and management.

Table 2. Manage Menu Functions

Menu Option	Description
Joins	This option opens the Joins Manager, where you can work with physical and logical joins.
Variables	This option opens the Variables Manager window, which you can use to create, edit or delete variables.
Projects	Opens the Project Manager window in which you can 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 192.

Tools

The Tools menu options allow you to set options for the Administration Tool, access the Time Series Wizard, and work with various management functions.

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 Systems Web site.
- About Administration Tool. Obtain information about the installed Administration Tool.
- About Siebel Analytics Server. Obtain information about the installed Siebel 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 3](#) presents the keyboard shortcuts you can use in the Administration Tool to perform frequent tasks.

Table 3. Keyboard Shortcuts

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

Icons and Symbols in the Administration Tool

Table 4 presents the icons and symbols used in the Administration Tool with an explanation of what each represents.

Table 4. 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 Siebel Analytics Server to use.
	A user database sign-on which is a hidden object visible only in the Query Repository dialog box.
	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.

Table 4. Icons and Symbols














Icon or Symbol	What It Represents
	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 key for a physical or logical table.
	A foreign key for a physical or logical table in the Joins Manager.
	A physical or logical complex join in the Joins Manager.
	A dimension in the Business Model and Mapping 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 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.

Table 4. Icons and Symbols











Icon or Symbol	What It Represents
	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.
	An expanded business model in the Business Model and Mapping layer that is not available for queries.
A	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_m	A measure definition.
	A user.
	A security group.
	An LDAP server.
	A group hierarchy.

Table 4. Icons and Symbols















Icon or Symbol	What It Represents
	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.
	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).

Table 4. Icons and Symbols

Icon or Symbol	What It Represents
	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.

Online Help in the Administration Tool

Most windows and dialog boxes have online help containing information to help you complete a task. To access a help file, click the help buttons in a window or dialog box.

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”](#)
- [“Opening a Repository in Online Mode” on page 61](#)

Opening a Repository in Offline Mode

Use offline mode to view and modify a repository while it is not loaded into the Siebel 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.

The Open Offline dialog appears.

- 3 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 Siebel Analytics Server. There are certain things you can do in online mode that you cannot do in offline mode. In online mode, you can:

- 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 the 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.

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 Siebel Analytics Server 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 Siebel 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 Siebel 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 65](#)
- [Using the Options Dialog Box—Repository Tab on page 67](#)
- [Using the Options Dialog Box—Sort Objects Tab on page 68](#)
- [Using the Options Dialog Box—Cache Manager Tab on page 69](#)
- [Using the Options Dialog Box—More Tab on page 69](#)

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.

Option	Description
Tile when resizing	Automatically tiles the layer windows when you resize the Administration Tool.
Wrap description text	Automatically wraps text in description boxes, eliminating the need to scroll to the right.
Display qualified names in diagrams	Displays qualified names in the Administration Tool diagrams, making it easier to identify sources.
Show Time Wizard menu always	Keeps the Time Series Wizard menu active for all business models, without checking to be sure that the business model meets requirements to use it.
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>

Option	Description
Check out objects automatically	(online mode only) When a user double-clicks an object to edit it, the object will be checked out automatically. If this option is not selected, the user will be prompted to check out the object before editing it.
Show row count in physical view	<p>Select this option to display 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 also not available for XML, XML Server, or multidimensional databases.</p>
Show Toolbar	Displays the toolbar.
Show Statusbar	Displays the status bar.

Using the Options Dialog Box—Repository Tab

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 116](#) and [“Setting Up Display Folders in the Business Model and Mapping Layer” on page 156](#).

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 which columns 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, choose a percentage.

The default window size is 75 percent.

For information about specifying the multiuser development directory, see [“Setting Up a Developer’s Machine for Analytics Multiuser Development” on page 195](#).

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 361](#).

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.

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
- Data sources which do not support the CountDistinct function, such as Microsoft Access, Microsoft Excel, and multidimensional data sources

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 dialog box from which you entered the Browse 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.

Administration Tool Basics

Using the Browse Dialog Box

Setting Up a Siebel Analytics Server Repository

4

This section contains an overview of the layers of an Analytics repository and provides instructions for creating a repository file, the first step in setting up a repository.

The Siebel Analytics Server stores metadata in repositories. The metadata contains information about the users or groups authorized to connect to the repository. You set up users, groups, access privileges, and query privileges using Siebel Analytics Server security. For more information, see [“Analytics Security Manager” on page 361](#).

The metadata also contains information about physical databases and other sources where the data is stored. It defines the logical business models which appear to users in the presentation layer.

Most windows and dialog boxes have online help containing information to help you complete a task. To access a help file, click the help buttons in a window or dialog box.

About Repository Structure in the Administration Tool

The Administration Tool employs a graphical user interface (GUI) that allows Siebel Analytics Server administrators to set up repositories. This section describes the components of the Administration Tool user interface.

A Siebel Analytics Server repository consists of three layers, depicted in the Administration Tool by three separate views. These layers are transparent to the end user, but are visible to the Siebel Analytics Server administrator. Each layer of the Administration Tool uses a tree structure (similar to the Windows Explorer) to expand objects into each of its components. Each of these layers is described in the next section.

Physical Layer in the Repository

The Physical layer contains information about the physical data sources. The most common way to populate 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 information about supported databases, see *System Requirements and Supported Platforms* on 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 details about connection pools, see [“Setting up Connection Pools” on page 91](#). For more information about setting up the Physical layer, see [Chapter 5, “Creating and Administering the Physical Layer in a Repository.”](#)

Business Model and Mapping Layer in the Repository

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 Siebel Analytics Server uses these definitions to pick the appropriate sources for each data request.

You can change the names of physical objects independent from corresponding business model object names and properties, and vice versa. Any changes in the underlying physical databases need not change the view of the tables in the business model that end user applications ultimately see and query—the business model to physical table mappings can change dramatically while remaining transparent to end user applications.

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 information about creating business model schemas, see [Chapter 2, “Data Modeling.”](#) For more information about setting up the Business Model and Mapping layer, see [Chapter 6, “Creating and Administering the Business Model and Mapping Layer in a Repository.”](#)

Presentation Layer in the Repository

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 7, “Creating and Maintaining the Presentation Layer in a Repository.”](#)

Process of Setting Up a Repository

Every repository contains a Physical layer, a Business Model and Mapping layer, and a Presentation layer. You can work on each layer at any stage in creating a repository. However, 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. After analyzing your business model needs, you can begin to create a repository. The usual order is to create the Physical layer first, the Business Model and Mapping layer next, and the Presentation layer last. You can set up security when you are ready to begin testing the repository.

Table 5 lists the tasks to set up a repository and the page number where each task is described.

Table 5. Repository Setup Checklist

Step #	Task
Step 1	“Creating a New Analytics Repository File” on page 78
Step 2	“Creating and Administering the Physical Layer in a Repository” on page 81
Step 3	“Creating and Administering the Business Model and Mapping Layer in a Repository” on page 127
Step 4	“Creating and Maintaining the Presentation Layer in a Repository” on page 165
Step 5	“Completing Setup and Managing Repository Files” on page 175

Creating a New Analytics Repository File

The first step in creating a repository is creating a 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** Select File > New from the Administration Tool menu 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 Analytics Server software installation folder.

The new repository is empty. The remaining steps in the repository setup process, as outlined in [Table 5 on page 78](#), tell you how to populate it.

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

After Creating a Repository File

The first step in setting up a repository is creating a repository file. The second step is creating the Physical layer in the Administration Tool. For more information, see [“Creating and Administering the Physical Layer in a Repository” on page 81](#).

Setting Up a Siebel Analytics Server Repository

Creating a New Analytics Repository File

Creating and Administering the Physical Layer in a Repository

5

Before you create the Physical layer of a repository, you need to create a repository file. For more information, see [Chapter 4, “Setting Up a Siebel Analytics Server Repository.”](#)

The Physical layer of the Administration Tool defines the data sources to which the Siebel 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 the Siebel Analytics Platform Server Extended product or the Siebel Analytics Server Stand-Alone product, you are authorized to add databases or 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. The method that you use to create the create the schema depends on your data source type. You can create the Physical layer manually (for a multidimensional data source) or import the schema (all other supported data sources).

- **Importing the physical schema to create the Physical layer.** You can import the physical schema for supported data source types (other than multidimensional). You can import schemas or portions of schemas from existing data sources.
- **Manually creating the Physical layer for a multidimensional data source data source.** When you want to connect to a multidimensional data source (cube structure), you need to manually create each element of the Physical layer. You cannot import these schemas at this time.

This section provides instructions for using the Administration Tool to create the Physical layer of a repository. 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.

This chapter includes the following topics:

- [Process of Creating the Physical Layer by Importing a Physical Schema](#)
- [Process of Creating the Physical Layer for a Multidimensional Data Source](#)
- [Setting Up Database Objects on page 88](#)
- [Setting up Connection Pools on page 91](#)
- [About Physical Tables on page 100](#)
- [Creating and Administering Physical Tables on page 102](#)
- [Creating Physical Layer Folders on page 114](#)
- [About Physical Joins on page 117](#)
- [Defining Physical Foreign Keys and Joins on page 120](#)
- [Using Database Hints on page 123](#)

Process of Creating the Physical Layer by Importing a Physical Schema

Importing the physical schema saves you time and effort by importing the structure for the physical layer. You can import schema for all supported data sources, except multidimensional. 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. 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 an Oracle 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 Siebel 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](#)
- [Setting Up Database Objects on page 88](#)
- [Setting up Connection Pools on page 91](#)

Importing a Physical Schema

You can import physical schema for supported data source types, except for multidimensional data sources. This task is part of “[Process of Creating the Physical Layer by Importing a Physical Schema](#)” on page 82.

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

- **Import.** 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 Siebel Analytics Server XML gateway, see [“Siebel Analytics Server XML Gateway Example” on page 394](#).
 - A comma separated value (CSV) or tab separated file can be imported using the TEXT/DDL connection type. You can create a file in Excel, save it in CSV or tab-separated format, and import the schema information into the repository.
 - To import from a text file, each line must contain the following information in the sequence shown:

```
catalog, schema, table, column, datatype, length, nullability
```

This text file cannot contain a heading row. If you want to omit elements such as schema, length, or nullability, you need to add a delimiter (comma or tab) to represent the omitted element. The following lines are examples of lines you might see in a CSV file in which the schema element was omitted:

```
StockQuotes, , stock, % Change, VARCHAR, 32, Y
```

```
StockQuotes, , stock, Change, VARCHAR, 32, Y
```

- **Import through Server.** Available in Online mode. Use this option when you want to use the Analytics Server connections to import schema. This feature allows an administrator to use the DSN's of the Siebel Analytics Server machine to import physical schema information. Connectivity software and duplicate DSN's do not have to reside on the administrator's machine and the Siebel 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.

- The TEXT/DDL connection type is not available for the Import through Server option.
- When it is running on a Unix platform, the Analytics Server does not support importing schema using an ODBC connection type.

NOTE: The Import from Repository menu option allows you import metadata objects from another repository. You can use this to import objects from all three metadata layers. For more information, see [“Importing from Another Repository” on page 179](#).

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.

To import a physical schema from an ODBC connection type

- 1** In the Administration Tool, select File > Import or Import through Server.
- 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.

To import a physical schema from a delimited file

- 1 In the Administration Tool, select File > Import or Import through Server.
- 2 In the Select Data Source dialog box, select the TEXT/DDDL connection type.
- 3 In the Select text file to import from dialog box, click Browse to locate and select the file, and then click OK.
- 4 In the Import dialog box, select each schema folder you want to import, and then click Import.

If you select only the highest level database object, you will import only the first folder.

NOTE: If a table or column listed in the text file already exists in the Physical layer, it will be overwritten during the import.

Process of Creating the Physical Layer for a Multidimensional Data Source

This section describes how you use the Administration Tool to add a multidimensional data source to the physical layer of a repository. Siebel Analytics uses XMLA to bring data from a multidimensional data source into the Siebel Analytics repository.

Each cube in a multidimensional data source is initially modelled as a single physical table. Cube-specific concepts such as hierarchies, levels, and properties are defined on the physical cube table.

When mapping to a multidimensional data source, you must build each element of the Physical layer, including hierarchies. No import utility is available at this time. To complete these tasks, you need to understand multidimensional data source concepts and be familiar with the structure of your data source.

NOTE: It is strongly recommended that you open your data source to use as a reference when creating the mappings between Siebel Analytics and your data source.

To help make sure that your queries return accurate data, use your data source to accomplish the following tasks:

- Understand how to fit your cube structure into the Siebel Analytics repository structure.
- Obtain information about each hierarchy that you want to access, so you can create an exact match in the Administration Tool. This includes the hierarchy name, its levels, and the properties defined at each level.
- Verify the spelling of all elements before creating the corresponding objects in the Administration Tool.

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

- [Setting Up Database Objects on page 88](#)

- [Setting up Connection Pools on page 91](#)
- [About Physical Tables on page 100](#)
- [Creating and Administering Physical Tables on page 102](#)

Setting Up Database Objects

Importing a schema automatically creates a database object for the schema and you need to set the database properties. For 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:

- [“Creating a Database Object in the Physical Layer”](#)
- [“Specifying SQL Features Supported by a Database” on page 90](#)

Creating a Database Object in the Physical Layer

The way you create a database object depends on whether you imported your schema or created it for a multidimensional data source.

- If you import the physical schema into the Physical layer, the database type is automatically assigned. During this import process, some ODBC drivers provide the Siebel Analytics Server with the database type. 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.

- If you create the physical schema in the physical layer of the repository for a multidimensional data source, you need to create one database in the physical layer for each cube, or set of cubes, that belong to the same catalog (or 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 XMLA tab of the Connection Pool dialog box.

NOTE: Note: if multiple cubes share dimensions, each shared dimension and hierarchy must be repeated for each physical cube table.

Before you begin, contact your system administrator to obtain vendor-specific information about your data source. Make sure to ask the following questions:

- Are your database entries case sensitive?
- Obtain values for setting up your connection pool. How should you connect to your data source? For example, you might obtain the URL to connect to the data source provider (this is input as the connection pool data source name) and the parameters necessary for connecting or accessing the data source (this is added in the XMLA tab of the Connection Pool dialog box).

To create a database object

- 1** In the Physical layer of the Administration Tool, right-click and select New Database from the shortcut menu.

Make sure that no object is selected when you right-click.

- 2** In the General tab, complete the necessary fields.
 - a** In the Name field, type the name you want to assign.
 - b** From the Database drop-down list, select the appropriate database type.
- 3** Click the Features tab to examine the SQL features supported by the database type specified in [Step 2](#).
- 4** Click the Display Folders tab to see display folders created by the administrator.

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 the following list to help you locate and specify SQL features:

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.
Default check box	A check box that identifies default SQL features. Default SQL features supported by the database type of the data source are automatically selected. It is strongly recommended that you do not disable the default SQL features.
Ask DBMS button	Used only if installing and querying a database that has no Features table. Allows you to query this database for Feature table entries. For more information, see “Changing Feature Table Entries Using Ask DBMS” on page 91 .

Field or Button	Description
Revert to Defaults button	Restores the default values.
Find button	Typing a string helps you locate a feature in the list.
Find Again button	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 Siebel Technical Support.

Setting up Connection Pools

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 Siebel Analytics Server and that data source.

The Physical layer in the Administration Tool contains at least one connection pool for each database. 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.

- When you create the physical layer by importing a schema for a data source, the connection pool is created automatically.
- When you create the physical layer for a multidimensional data source, you need to create the connection pool. Each cube can have only one connection pool.

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 Siebel Analytics Server allocates shared memory for each connection upon server startup. This raises the number of connections and increases Siebel Analytics Server memory usage.

This section includes the following topics:

- [“Creating a Connection Pool for All Data Sources” on page 93](#)
- [“Setting Up Additional Connection Pool Properties for an XML Data Source” on page 97](#)
- [“Setting Additional Connection Pool Properties for a Multidimensional Data Source” on page 99](#)

Table 6 lists several supported connection types (APIs) with a description of each interface.

Table 6. Connection Type (API) Descriptions

API Name	Description
Default	Uses the API specified in the data source. If the data source is configured using an ODBC 2.0 driver, ODBC 2.0 is used. If the data source is configured using OCI for Oracle8, OCI8 is used, and so on. The connection pool shows what API is used as the default.
ODBC 2.0	Uses the open database connectivity (ODBC) standard Version 2.0 API. ODBC 2.0 offers connectivity to many common databases and OLAP tools.
ODBC 3.5	Uses the open database connectivity (ODBC) standard Version 3.5 API. Recommended for internationalization. ODBC 3.5 is strongly recommended over ODBC 2.0 at all times.
OCI 7	Uses the Oracle Call Interface (OCI) for Oracle7.
OCI 8	Uses the Oracle Call Interface (OCI) for Oracle8.
OCI 8.1	Uses the Oracle Call Interface (OCI) for Oracle 8i and 9i. Recommended for internationalization.
DB2 CLI	Uses the Call Level Interface (CLI) for DB2.
DB2 CLI Unicode	Uses the unicode Call Level Interface (CLI) for DB2. Recommended for internationalization.
XML	Uses the Siebel Analytics XML gateway to access XML data sources.

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

Creating a Connection Pool for All Data Sources

You must create a database object before you create a connection pool. After creating a database object, you create or edit a connection pool in the Physical layer of the Administration Tool.

To create a connection pool and set up General properties

- 1** In the Physical layer of the Administration Tool, right-click a database and select New Object and then Connection Pool.
- 2** In the Connection Pool dialog box, in the General tab, complete the fields using information in [Table 7](#).

Table 7. Connection Pool General Properties

Field or Button	Description
Name	The name for the connection pool. If you do not type a name, the Administration Tool generates a name.
Permissions	Click to access the Permissions dialog box. You can use the Permissions dialog box to assign 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.
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. <ul style="list-style-type: none">■ If the call interface is XML, the XML tab is available but options that do not apply to XML data sources are not available.■ If the call interface is XMLA, the XMLA tab is available but options that do not apply to multidimensional data sources are not available.
Maximum connections	The maximum number of connections allowed for this connection pool.

Table 7. Connection Pool General Properties

Field or Button	Description
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>
Data source name	<p>The drop-down list shows the User and System DSNs configured on your system. A data source name 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.</p>
Shared logon	<p>Select the option Shared logon 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 Siebel Analytics 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>
Enable connection pooling	<p>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.</p>

Table 7. Connection Pool General Properties

Field or Button	Description
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>
Execute queries asynchronously	<p>Indicates whether the data source supports asynchronous queries. Asynchronous queries allow query cancellations to be sent to the database through the same connection while the query is executing. If this option is not selected, query cancellations might not propagate down to the underlying database.</p>
Execute on Connect	<p>Allows the Siebel 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.</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>
Isolation level	<p>For ODBC and DB2 gateways, sets the transaction isolation level on each connection handle to the back-end database.</p>

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, and then select Connection Pool.
- 2 In the Connection Pool dialog box, click the XML tab.
- 3 Complete the fields, using [Table 8](#) as a guide.

Table 8. XML Connection Pool Properties

Property	Description
Search script	Used for XML Server data source.
URL refresh interval	<p>The refresh interval for XML data sources is analogous to setting cache persistence for database tables. The URL refresh interval is the time interval after which the XML data source is to be re-queried directly rather than using results in cache. The default setting is infinite, meaning the XML data source will never be refreshed.</p> <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.

Table 8. XML Connection Pool Properties

Property	Description
URL loading time out	<p>Time-out interval for queries. The default is 15 minutes.</p> <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 of the interval.
Maximum connections	Default is 10.
Query input supplements Header file/Trailer file	Used for XML Server data source.
Query output format	XML is the only choice for an XML data source. Other choices are available for an XML Server data source.
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 will apply it to all the XML physical tables in the connection pool. You can also specify an XSLT file on a per-table basis on the XML tab of the Physical Table object, which overrides what was specified in the connection pool.</p>
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 */BOOK[not(PRICE > '200')].</p>

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 Additional Connection Pool Properties for a Multidimensional Data Source

Use the XMLA tab of the Connection Pool dialog box to set additional properties for a multidimensional data source.

To set connection pool properties for a multidimensional data source

- 1 Right-click the multidimensional database and select New Object and then select Connection Pool.
- 2 In the Connection Pool dialog box, click the XMLA tab.
- 3 Complete the fields, using [Table 9](#) as a guide.

Table 9. XMLA Connection Pool Properties

Property	Description
URL	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 which comes with the Microsoft XML for Analysis SDK installation. For example, the URL might look like the following: http://SDCDL360i101/xmla/msxisap.dll
Data Source Information	Use your vendor-specific information 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 <code>Provider-MSOLAP;Data Source-local</code> . If using v1.1, then it should be <code>Local Analysis Server</code> .
Catalog	See your database administrator for the name of the back end data source which contains the cubes that you want to access.

Table 9. XMLA Connection Pool Properties

Property	Description
URL loading time out	Time-out interval for queries. The time to return results from the query or request from the multidimensional data source. The default timeout is 30 seconds. Set values as follows: <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 of the interval.
Use session	Default is Off (not checked). Controls whether queries go through a common session. Consult your multidimensional data source administrator to determine whether this option should be enabled.

About Physical Tables

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 Siebel 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 261](#). Creating virtual tables can provide the Siebel 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 10](#) provides a description of the available object types.

Table 10. 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 information about database hints, see “Using Database Hints” on page 123 .
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 information about database hints, see “Using Database Hints” on page 123 . 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.

Table 10. Object Type Descriptions for Physical Tables

Object Type	Description
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>
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 Administering Physical Tables

For all data sources, you can define general properties, columns, a primary key, and foreign keys. However, because you cannot import tables for multidimensional data sources, you must create each physical table cubes and define its properties.

Physical Tables for Multidimensional Data Sources

Importing multidimensional physical schema is not supported at this time. For multidimensional data sources, you need to manually create all elements of a physical table cube. This includes columns, a primary key, foreign keys, and hierarchies.

Each cube from a multidimensional data source is modelled 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, see [“Icons and Symbols in the Administration Tool” on page 56](#).

NOTE: 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 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.

For supported data sources (not multidimensional), you can import the physical schema. If the keys are defined in the database and the table definitions are imported with their columns and keys, the table properties are configured automatically.

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

- [Creating and Administering General Properties for a Physical Table on page 103](#)
- [Creating and Administering Columns and Keys in a Physical Table on page 105](#)
- [Adding and Administering Hierarchies in the Physical Layer for a Multidimensional Data Source on page 109](#)
- [Setting Physical Table Properties for an XML Data Source on page 114](#)

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 database and select New Object and then Physical Table.
 - To create a physical cube table for a multidimensional data source, right-click the database, select New Object, and then Physical Cube Table.
 - To edit an existing physical table, double-click the physical table icon for the physical table that you want to edit.
- 2 In the selected Physical Table dialog box, complete the fields using “[Physical Table General Properties](#)” as a guide.

Table 11. Physical Table General Properties

Property	Description
Name	The administrator assigns a name to the new table.
Repository Object Name	The name of the table and its associated path. <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 Siebel 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 11. 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 information about the cache polling table, see “Troubleshooting Problems with an Event Polling Table” on page 309.</p>
Object Type	<p>See Table 10 on page 101 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 select Delete.

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 106](#)
- [“Specifying a Primary Key for a Physical Table” on page 108](#)
- [“Deleting a Physical Column For All Data Sources” on page 109](#)

Creating and Editing a Column in a Physical Table

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

If you are connecting to a multidimensional data source, columns cannot be imported. Therefore you need to manually create all elements of a cube table, including columns.

NOTE: Except when stated otherwise, the characteristics and behavior of a physical cube column is 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 select New Physical Column from the shortcut menu.
 - To create a physical cube column for a multidimensional data source, right-click a physical cube table, select New Object, Physical Cube Column.

- To edit an existing physical column, double-click the icon for the physical column that you want to edit.
- 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 datatype for the physical column.

For multidimensional data sources, select DOUBLE for measures or VARCHAR for all other columns.
- 4** If applicable, specify the length of the datatype.

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 datatype list indicates the datatype 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 nullable value to a not 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 datatype values, however; setting the values to ones that are incorrect in the underlying data source might cause unexpected results.

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

- 6 (Optional) The External Name field will store and display the fully qualified name of a column (attribute) in an XML document.

An external name is required if the same name (such as STATE) is used in multiple hierarchies.

NOTE: A new physical cube column is created as a measure by default. To change this, see [“Adding and Administering Hierarchies in the Physical Layer for a Multidimensional Data Source”](#) on page 109.

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 To add a column to the Columns list:
 - a Click Add.
 - b In the Browse dialog box, select the column, and then click OK.

The column you selected appears in the Columns list of the Physical Key dialog box.
- 7 (Optional) Type a description for the key.

Deleting a Physical Column For All Data Sources

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

NOTE: A physical cube column belongs to a physical cube table. If you delete a cube column from the physical layer, the column data will not be available for queries.

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 select Delete.

Adding and Administering Hierarchies in the Physical Layer for a Multidimensional Data Source

When you create a new cube column, it is created as a measure by default. 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.

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 will not typically return the correct results.

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

- [Adding a Hierarchy to a Physical Cube Table](#)
- [Verifying Hierarchy Levels on page 111](#)
- [Adding or Removing a Cube Column in an Existing Hierarchy on page 112](#)
- [Removing a Hierarchy From a Physical Cube Table on page 113](#)
- [Associating a Physical Cube Column With a Hierarchy Level](#)

Adding a Hierarchy to a Physical Cube Table

Each 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 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, you must 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.

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.
- 3** In the Hierarchies tab, click Add.
- 4** In the Hierarchy dialog box, in the Level tab, type a name for the hierarchy in the Name field, and click Add.
- 5** In the Physical Level dialog box, in the Columns tab, type a name for the level in the Name field and click Add.
- 6** In the Browse dialog box, in the Name list, locate the columns that you want to add to the hierarchy.

For more information about the Browse dialog box, see [“Using the Browse Dialog Box” on page 72](#).

- 7 Add columns in top-down order as explained in the following steps:

CAUTION: You must add the columns to your hierarchy, starting with the highest level of your hierarchy, so that each level appears in the correct sequence. If the columns are not in the correct sequence, click Up or Down to correct the order of the levels. Using the correct hierarchical sequence allows your queries to return accurate information and avoids errors.

- a Add the column for the highest level in your hierarchy by clicking that column and clicking Select.
 - b Select subsequent columns in top-down order and click Select.
- 8 In the Physical Level dialog box, click OK.
In the Physical Level dialog box, the Columns tab lists all the columns selected for that level.
 - 9 In the Hierarchy dialog box, to change the order of the hierarchies, select a hierarchy and click Up or Down.
In the Hierarchy dialog box, the Levels tab lists all the defined levels for the selected hierarchy. You must select a hierarchy for the buttons to be available.
 - 10 In the Hierarchy dialog box, click OK.

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 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, in the Level tab, verify the levels are correct.

- 5** If you need to reorder the hierarchy levels, select a level and click Up or Down to correct the order of the levels.
- 6** When the levels are correct, 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 and from the cube table in the Physical layer.

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, making sure that they are in the correct sequence.
 - 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.

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 RDBMS 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 must exist in the multidimensional data source and 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, if you set 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 97](#).

Creating Physical Layer Folders

This section contains the following topics:

- [Creating Catalog Folders](#)
- [Creating Schema Folders on page 115](#)
- [Using a Variable to Specify the Name of a Catalog or Schema on page 115](#)
- [Setting Up Display Folders in the Physical Layer on page 116](#)

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 select 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 select 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 Siebel 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 67](#).

NOTE: Deleting objects in the display folder only deletes 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 Siebel Analytics Server.

Multi-Database Joins

A multi-database join is defined as a table in one database that logically joins to a table in another database. 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 Siebel Analytics Server. For information about the Physical Table Diagram, see [“Defining Physical Joins in the Physical Diagram”](#) on page 121.

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 10](#) 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).

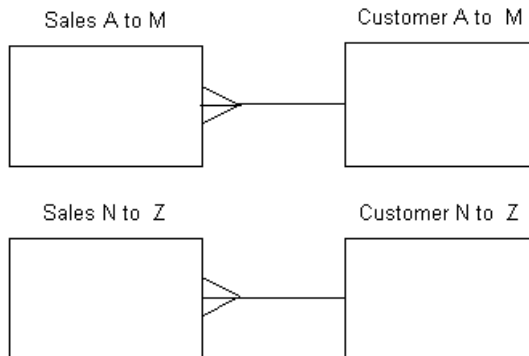


Figure 10. Fragmented Tables Example

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 11](#).

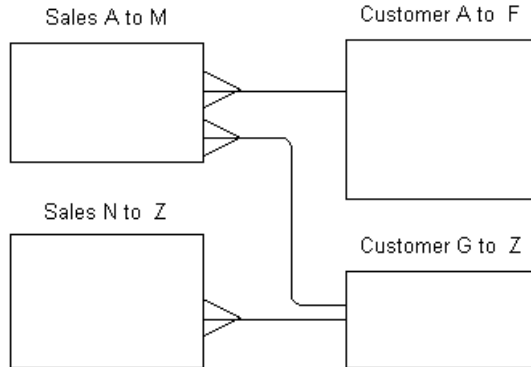


Figure 11. Joins for Fragmented Tables Example

TIP: Avoid adding join conditions where they are not necessary (for example, between Sales A to M and Customer N to Z in [Figure 10](#)). 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 121 and “[Creating and Administering Columns and Keys in a Physical Table](#)” on page 105.

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 and logical complex 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 120](#)
- [Defining Physical Joins in the Physical Diagram on page 121](#)

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 information about database hints, see [“Using Database Hints” on page 123](#).
- 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 12](#).

To display the Physical Table Diagram

- 1 In the Administration Tool, in the Physical layer, right-click to select a table.
- 2 In the first shortcut menu, choose Physical Diagram.
- 3 In the second shortcut menu, choose an options from [Table 12](#).
- 4 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 12](#).

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

Table 12. 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 Siebel 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 162](#).

- 6 If appropriate, specify a database hint.

For information about database hints, see [“Using Database Hints” on page 123](#).

- 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. Siebel 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 13 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 13. 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 Siebel Analytics Server. For 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:

```
index(ORDER_ITEMS, FAST_INDEX)
```

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:

```
leading(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 Siebel Analytics Server inserts the comment markers when the hint is executed.

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

6

This section is part of the Process of setting up a repository. For more information, see [“Process of Setting Up a Repository” on page 78](#).

After creating all of the elements of the Physical layer, you can drag and drop tables and columns from the Physical layer to the Business Model and Mapping (logical) layer. For more information about creating the Physical Layer, see [“Creating and Administering the Physical Layer in a Repository” on page 81](#).

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.

See the following topics for information about creating the business model and logical objects:

- [Creating Business Model Objects on page 128](#)
- [Creating and Administering Logical Tables on page 129](#)
- [Creating and Administering Logical Columns on page 134](#)
- [Creating and Administering Logical Table Sources \(Mappings\) on page 137](#)
- [About Dimensions and Hierarchical Levels on page 143](#)

- [Process of Creating and Administering Dimensions on page 143](#)
- [Setting Up Display Folders in the Business Model and Mapping Layer on page 156](#)
- [Defining Logical Joins on page 157](#)

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.

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 129](#)
- [“Adding or Editing Logical Table Sources” on page 131](#)
- [“Specifying a Primary Key in a Logical Table” on page 133](#)
- [“Editing Foreign Keys for a Logical Table” on page 134](#)

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 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 157](#) and [“Defining Logical Joins with the Business Model Diagram” on page 160](#).

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 information about bridge tables, see [“Bridge Tables to Model Many-to-Many Relationships” on page 46](#).

- 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. 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.

To add or edit a new logical table source

- 1** In the Business Model and Mapping layer, double-click a table.
- 2** In the Logical Table dialog box, select the Sources tab.
- 3** In the Sources tab, perform one of the following steps:
 - To add a table source, click Add.
 - To edit an existing table source, select a source and click Edit.
- 4** In the Logical Table Source dialog box, in the General tab, complete the fields.
- 5** Click the Column Mapping tab and select one of the following check boxes or click New column:
 - Show mapped columns. Shows current mapping for the table.
 - Show unmapped columns. Shows columns available for mapping.
- 6** Set up mappings for the table source.
- 7** Click the Content tab and complete the fields using [Table 14](#) as a guide.

8 Click OK.

Table 14. Content Tab Fields for Logical Table Source

Field	Description
Aggregation content, group by	How the content is aggregated.
Show mapped/Show unmapped	Select check box to see a list of mapped or unmapped columns. Clicking the Dimension header toggles the sort between ascending and descending order.
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 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.)

Table 14. Content Tab Fields for Logical Table Source

Field	Description
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.
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	Not currently used.

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.

NOTE: Logical dimension tables must have a logical primary key. Logical keys can be composed of one or more than one logical columns. Logical keys are optional for logical fact tables, although the administration tool may warn you if they do not exist.

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** To add a column to the Columns list:
 - a** Click Add.

- b** In the Browse dialog box, select the column, and then click OK.

The column you selected appears in the Columns list of the Logical Key dialog box.

- 5** (Optional) Type a description for the key.
- 6** Click OK.

Editing Foreign Keys for a Logical Table

If you need to add or change foreign keys, use the Foreign Keys tab to identify or edit the foreign keys for a logical table.

NOTE: It is recommended that you do not have foreign keys for logical tables.

To edit a foreign key 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 Keys tab.
- 3** In the Keys list, select a key.
- 4** Click Edit.
- 5** Select or clear the check box for the appropriate columns, and click OK.

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, are 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 a Logical Column](#)
- [Setting Default Levels of Aggregation for Measure Columns on page 136](#)
- [Associating an Attribute with a Level in Dimension Tables on page 136](#)

Creating 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.

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** 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.

- 6 (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.

- 7 Make changes and click OK.

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, leave the aggregation Type as Simple and 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 Level in Dimension Tables

Attributes can be associated with a 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 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 149](#).

To associate this measure with a level in a dimension

- 1 Click in the white space in the Levels list next to the dimension from which you want to select a level.
- 2 Select the level.
- 3 Repeat this process to associate this measure with other levels in other dimensions.

To remove an association

- 1 Click the Delete button next to the level association you want to remove.
The level is removed from the Levels list.
- 2 Repeat this process to remove associations with levels in other dimensions.

Creating and Administering Logical Table Sources (Mappings)

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 aggregate navigation, see [“Setting Up Aggregate Navigation” on page 249](#).

This section includes the following topics:

- [Creating or Removing a Logical Table Source on page 138](#)
- [Defining Physical to Logical Table Mappings on page 139](#)
- [Defining Content of Sources on page 141](#)

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, type a name for the logical table source.
- 3** Click Add.
- 4** 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.

- 5** 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.
- 6** To add tables to the table source, select the tables in the Name list and click Select.

To remove a table as a 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.
- 3** In the tables list, select the table you want to remove and click Remove.
- 4** After removing the appropriate tables, click OK.

Defining Physical to Logical Table 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 datatype 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, 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 12 on page 140](#).
- 4** In the Physical Table list, 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.

- 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.

- To remove a column mapping, click the delete button next to the Physical Table cell.
- After you map the appropriate columns, click OK.

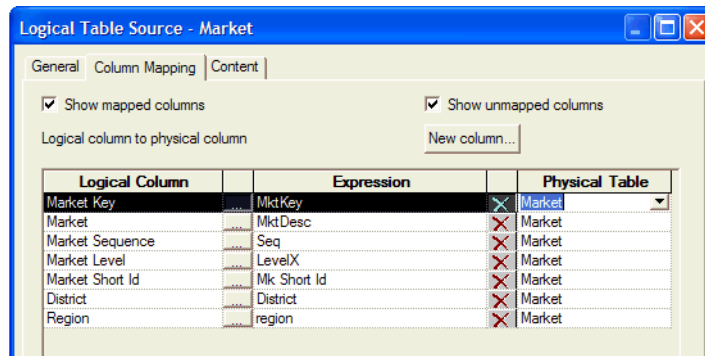


Figure 12. Logical Table Source Dialog Box

To remove a column mapping

- In the Business Model and Mapping layer, right-click a logical table and choose New Object > Logical Table Source.
- In the Logical Table Source dialog box, click the Column Mapping tab.
- In the Column Mapping tab, maximize or enlarge the dialog box to show all the contents, as shown in [Figure 12](#).
- To remove a column mapping, click the delete button next to the Physical Table cell.
- 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 Data Type tab.
- 3 In the Data Type tab, in the Logical Table Source list, select a source and click Unmap.
- 4 Click OK.

Defining Content of Sources

To use a source correctly, the Siebel Analytics Server has to know what each source contains in terms of the business model you are defining. 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 any Where clauses to limit the number of rows returned. For more information about aggregate table columns, see [“Specify the Aggregate Levels for Each Source” on page 250](#).

To create table content definitions

- 1 Access the Content tab of the Logical Table Source dialog box.
- 2 If a logical source is an aggregate table and you have defined logical dimensions, do the following:
 - a Select Logical Level from the Aggregation content drop-down list.
 - b In the Logical Level pane, select the appropriate level for the dimension.

You should specify a 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.
- 3 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 drop-down list.
- 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 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.

- 4** 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 to access the Fragmentation Content Expression Builder, where you specify content in terms of existing logical columns. For more information about fragmentation content, see [“Specify Fragmentation Content” on page 253](#).

- 5** Select the option This source should be combined to specify that this source is combined with other sources.

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

- 6** (Optional) Specify Where clause filters in the Where Clause Filter window to limit the number of rows the source uses in the resultant table.

- a** Click the Expression Builder button to open the Physical Where Filter Expression Builder.

- b** Type the Where clause and click OK.

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

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 levels represent the organizational rules, and reporting needs required by your business. They provide the structure (metadata) that the Siebel 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:

- Aggregate navigation
- Configure level-based measure calculations (see [“Level-Based Measure Calculations Example”](#) on page 149)
- Determine what attributes appear when Siebel 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 levels, and each level has one or more attributes (columns) associated with it.

NOTE: The concept of dimensions for a multidimensional (XMLA) 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 144](#)
- [Creating Dimensions Automatically on page 151](#)

Creating Dimensions

When creating a dimension, each dimension can be associated with attributes (columns) from just one logical dimension table (plus level-based measures from the 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 levels. The recommended sequence for creating levels is to create a grand total level and then create child levels, working down the 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 levels. Any column that you do not associate with a 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 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 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 level may have more than one level key. When that is the case, specify which key 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 level should have one level key that will be displayed when a Siebel Answers or 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 which 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:

To create dimension levels, perform the following tasks:

- [Creating a Logical Level in a Dimension on page 146](#)
- [Associating a Logical Column and Its Table With a Dimension Level on page 148](#)
- [Identifying the Primary Key for a Dimension Level on page 148](#)
- [Adding a Dimension Level to the Preferred Drill Path on page 149](#)
- [Level-Based Measure Calculations Example on page 149](#)
- [Grand Total Dimension Hierarchy Example on page 151](#)

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.

To define general properties for the dimension level

- 1** Specify a name for the level, or accept the default.
- 2** Specify the number of elements that exist at this 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 level to another should be accurate.

- 3** If this 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.
- 4** To define child levels, click Add and perform the following steps:
 - a** Select the child 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.
- 5** (Optional) Type a description of the level.

Associating a Logical Column and Its Table With a Dimension Level

After you create all levels within a dimension, you need to drag and drop one or more columns from the dimension table to each 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 level to be associated with that logical column, you can drag a column from one 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 Open the Logical Level dialog box and click the Keys tab.
- 2 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.

- 3 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 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 level from a drill path or reorder a level in the drill path.

To add a level to the preferred drill path

- 1 Click the Add button to open the Browse dialog box, where you can select the levels to include in the drill path. You can select 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.

Level-based measures allow a single query to return data at multiple levels of aggregation. They are also useful in creating share measures, which 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.

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

Process of Creating and Administering Dimensions

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 underlyingly 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.

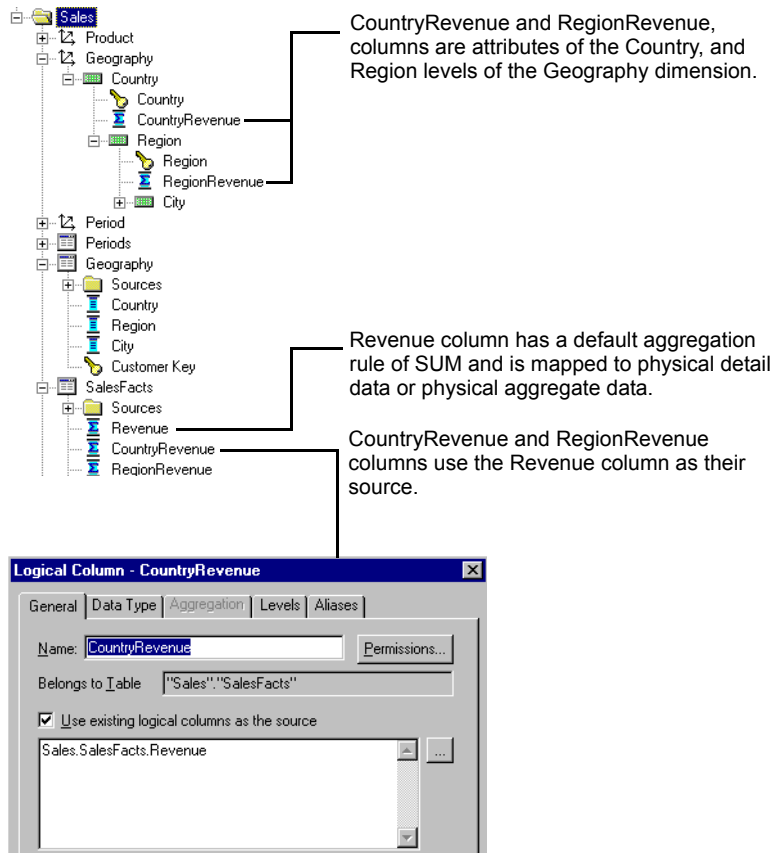


Figure 13. Example Business Model in the Business Model and Mapping Layer

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 which is defined with a default aggregation rule of Sum. You can then create a logical column, AllProductRevenue, which 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 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 levels. These additional level names use the logical column names of these key columns.
 - The order of joins determines the hierarchical arrangement of the levels. The level keys of these new 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 levels, counting the grand total and detail levels.

- The key of the table in source B (which 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 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 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.

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.

If a dimension with joins and levels has already been created, Create Dimension will not appear on the right-click menu.

- 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 Siebel 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 Aggregate Navigation](#)” on page 249.

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 to open the Expression Builder - Aggregate dialog box.
- 7** From the Formula drop-down list, select the aggregation to perform over the dimension or use the Expression Builder to configure the aggregation.
- 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 67](#).

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

Specifying the logical table joins is required so that the Siebel 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 the 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.

To create logical joins, perform the following tasks:

- [Defining Logical Joins with the Joins Manager on page 157](#)
- [Defining Logical Joins with the Business Model Diagram on page 160.](#)

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”](#)
- [“Creating a Logical Complex Join” on page 159](#)

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 Siebel 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 162](#).

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 Siebel 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 162](#).

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 Siebel 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 162](#).

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 Siebel Analytics Server processes cross-database joins when one table is very small and the other table is very large. When a driving table is specified, Siebel 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.

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.

You can specify a driving table in the Logical Foreign Key dialog box (accessed by editing an existing key or clicking the New button on the Foreign Keys tab of the Logical Table dialog), or in the Logical Join dialog box. This option also appears in the Physical Join dialog box (accessed by editing an existing key or clicking the New button on the Foreign Keys tab of the Physical Table dialog) but it is not available for you to select, because the Siebel Analytics Server implements driving tables only in the Business Model and Mapping Layer.

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.

To specify a driving table for logical joins

- 1** Access the appropriate dialog box.

- 2 Select the table from the Driving drop-down list, and then select an applicable cardinality.

CAUTION: 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. 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. To be safe, only specify driving tables when the driving table is extremely small - less than 1000 rows.

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.

Creating and Maintaining the Presentation Layer in a Repository

7

This section is part of the process of setting up a repository. For more information, see [“Process of Setting Up a Repository” on page 78](#). 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 127](#).

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.

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 Analytics Web users. They appear as catalogs to client tools that use the Siebel 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 17, “Security in Siebel Analytics.”](#)

Rename Presentation Tables to User-Friendly Names

Presentation tables can have a name different from logical tables in the Business Model and Mapping layer. You can rename a presentation table using the Presentation Table dialog box.

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 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”](#)
- [“Working with Presentation Tables” on page 171](#)
- [“Working with Presentation Columns” on page 172](#)
- [“Using the Alias Tab of Presentation Layer Dialog Boxes” on page 174](#)

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.

Presentation Catalog Dialog Box

The Presentation Catalog dialog box has three tabs: General, Presentation Tables, and Aliases. The functionality provided in each tab is described in [Table 15](#).

Table 15. 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 from the shortcut menu.

The Presentation Catalog dialog box appears.

- 2** In the General tab, type a name for the presentation catalog.
- 3** Click the Permissions button to open the Permissions dialog box, where you can assign user or group permissions to the catalog folder.

For information about assigning permissions to a presentation catalog, see [“Setting Permissions for Repository Objects”](#) on page 70.

- 4** Select a Business Model from the drop-down list.

After the Presentation Catalog has columns, the drop-down list becomes inactive.

- 5** To expose the logical keys to other applications, select the option Export logical keys.

This 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 Siebel Analytics Server uses the joins defined in the repository.

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.

An implicit fact column is a column that 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.

To delete a presentation table

- 1 Click the Presentation Tables tab.
- 2 Click the table you want to remove and click the Remove button, or press the Delete key.

A confirmation message appears.

- 3 Click Yes to remove the table, or No to cancel the removal.

To reorder a table in the Presentation layer

- 1 Click the Presentation Tables tab.
- 2 In the Name list, select the table you want to reorder.
- 3 Use the drag-and-drop feature to reposition the table, or click the Up and Down buttons.

To sort all tables in the Presentation layer in alphanumeric order

- 1 Click the Presentation Tables tab.
- 2 Click on Name column heading.

This toggles the sort between ascending and descending alphanumeric order.

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 16](#).

Table 16. 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 information about assigning permissions to a presentation table, see [“Setting Permissions for Repository Objects” on page 70](#).

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

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 two tabs: General and Aliases. The functionality provided in each tab is described in [Table 17](#).

Table 17. Presentation Column Dialog Box

Tab	Comment
General	Use this tab to create or edit a presentation columns.
Aliases	Use this tab to specify or delete an alias for a presentation column.

To create or edit a presentation column

- 1 Right-click a presentation table in the Presentation layer, and then select New Presentation Column from the shortcut menu.

The Presentation Column dialog box appears.

- 2 To specify a name that is different from the Logical Column name, do the following:
 - a Type a name for the column.
 - b Clear the Use Logical Column check box.

- 3** 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 is displayed in the Logical Column Name field.
- 4** Click the Permissions button to open the Permissions dialog box, where you can assign user or group permissions to the column.
- 5** If you want to examine or change the properties of the logical column in the Business Model and Mapping layer, click the Edit button to open the Logical Column dialog box.
- 6** Click OK when you are finished to return to the Presentation Column dialog box.
- 7** (Optional) Type a description of the presentation column.
- 8** To define any aliases for the logical column, click the Aliases tab.

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** Navigate to a Presentation Layer dialog box.
- 2** Click the Aliases tab.
- 3** Type the text string to use for the alias.

To delete an alias

- 1** In the Alias window, select the alias you want to delete.
- 2** Click the Delete button.

This section is part of the process of setting up a repository. For more information, see [“Process of Setting Up a Repository” on page 78](#). 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 179](#)
- [Querying and Managing Repository Metadata on page 180](#)
- [Constructing a Filter for Query Results on page 184](#)
- [Comparing Repositories on page 186](#)
- [Merging Siebel Analytics Repositories on page 188](#)
- [Creating an Analytics Multiuser Development Environment on page 191](#)
- [Making Changes in an Analytics Multiuser Development Environment on page 194](#)

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 176](#)
- [Add an Entry in the NQSConfig.INI File on page 177](#)

- [Create the Data Source on page 177](#)
- [Start the Siebel Analytics Server on page 178](#)
- [Test and Refine the Repository on page 178](#)
- [Publish to User Community on page 178](#)

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 points out a condition that the administrator should be aware of but which could be correct. See [“Checking the Consistency of a Repository or a Business Model” on page 63](#).

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 Siebel Analytics Server to load the repository into memory upon startup. The NQSConfig.INI file is located in the Config folder in the Siebel Analytics Server installation folder.

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;
```

The logical name is the name end users have to configure when configuring a DSN in the Siebel 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 on the NQSConfig.INI file, see *Siebel Analytics 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 Analytics Web has the same relationship to the Siebel Analytics Server as any other client application.

The steps to create a new data source are given in [Chapter 14, “Connectivity and Third-Party Tools.”](#) You can create standard data sources, and data sources that will participate in a cluster.

Start the Siebel Analytics Server

When you start the Siebel Analytics Server, the repositories specified in the NQConfig.INI file are loaded and become available for querying. For detailed information on starting the server, see [“Starting the Siebel Analytics Server” on page 265.](#)

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.

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 Siebel 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.

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.

When opening a repository online, you might want to select the option Load All Objects to load all repository objects during initial connection. This may slow the initial connection time but it speeds the execution of other tasks.

- 2** In the Repository Import Wizard, select a repository either by its file name or, if it is being used by a Siebel Analytics Server, by the Siebel Analytics ODBC DSN that points to the desired repository on that server.
- 3** Type the appropriate Siebel 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 18 on page 180](#) 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 which 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 18. Categories of Repository Objects to Import

Category	Description
Projects	Displays all repository objects that you have chosen to update and synchronize.
Catalogs	The Add with Children button is 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	The Add, Add with Children, and Add with Parents buttons are active. (You can view information about group membership from the Security Manager.)
Users	The active buttons depend on the user selected in the left pane.
Variables	Defined system and session variables appear in the left pane. The Add and Add with Parents buttons are active.
Initialization Blocks	When selected, the Add with Children button is active.

Querying and Managing Repository Metadata

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 From the Administration Tool menu bar, choose Tools > Query Repository.
- 2 In the Query Repository dialog box, complete the field on which you want to search.

For descriptions of each field and button in the Query Repository dialog box, see [Table 19 on page 182](#).

- 3 Click Query.
- 4 To save a repository query to an external file, click Save.
- 5 In the Save As dialog box, choose a type of file and an Encoding value.
- 6 To add additional columns of information to the results, click Add Columns.
- 7 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.

- 8 In the Save as dialog box, type a file name and select a Save as type.
- 9 Click Save.

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.

To create and copy a UDML report

1 From the Administration Tool menu bar, choose Tools > Query Repository.

2 In the Query Repository dialog box, complete the field on which you want to search.

For descriptions of each field and button in the Query Repository dialog box, see [Table 19 on page 182](#).

3 Click Query.

4 Select one or more objects, and then click UDML.

5 To copy the results to the Windows clipboard, on the UDML window, click Copy.

6 Open a program such as Windows Notepad and paste the clipboard information.

For more information about using a UDML report, see [“Execute UDML” on page 237](#).

Table 19. Query Repository Fields

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> .

Table 19. Query Repository Fields

Field or Button	Description
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 184.
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.
UDML	Universal Data Modeling Language. The internal language of the Siebel Analytics metadata. After executing a query, use this button to display the UDML for one or more selected object.
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.

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, which lets you delete the highlighted filter.

To access the Query Repository Filter dialog box

- 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 (when applicable).

To construct a filter

1 Access the Query Repository Filter dialog box.

2 Click in the right side of the blank row of the Item column.

A drop-down list displays the items by which you can filter for the object you selected in the Query Repository dialog box.

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 In the Query Repository dialog box, select Database from the Type drop-down list, and then click Filter.

The Query Repository Filter dialog box appears.

2 Click in the right side of the blank row of the Item column.

A drop-down list displays the items by which you can filter.

3 Select Related to from the Item drop-down list.

4 Select = in the Selection column to the right of the Item list.

5 Click in the Selection column to the right of the Value list.

The Browse dialog box appears.

- 6 Select the business model by which you want to filter.

Your selection appears in the Value list.

- 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 In the Query Repository dialog box, select Presentation Column from the Type drop-down list, and then click Filter.

The Query Repository Filter dialog box appears.

- 2 Click in the right side of the blank row of the Item column.

A drop-down list displays the items by which you can filter.

- 3 Select Column from the Item drop-down list.

- 4 Select = in the Selection column to the right of the Item list.

- 5 Click in the Selection column to the right of the Value list.

The Browse dialog box appears.

- 6 Select the column by which you want to filter.

Your selection appears in the Value list.

- 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 column becomes active, which lets you 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 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 from Siebel Systems.

For information about merging the contents of your customized Siebel Analytics applications repository with that of a new version of the repository, see [“Merging Siebel Analytics Repositories” on page 188](#).

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 60](#).

- 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:

Table 20. Change Field for Comparing and Merging Repositories

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 buttons on the right side of the Compare Repositories dialog box provide the following functionality:

Table 21. Functions Available When Comparing and Merging Repositories

Button	Functionality
Diff	Differences between the current repository and the original repository. For definitions of the values in the Change field, see Table 20 .
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.
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, enable the Turn Off Compare Mode. Not available when merging repositories.
Save	Saves a list of the differences between the two repositories.

Table 21. Functions Available When Comparing and Merging Repositories

Button	Functionality
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 that will be occur during update and synchronization.
UDML 1	Universal Data Modeling Language information for objects in the original repository.
UDML 2	UDML information for objects in the current repository.
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.

To enable the Turn Off Compare Mode

- From the Administration Tool toolbar, select File > Turn Off Compare Mode.

Merging Siebel Analytics Repositories

This section is for organizations that use the Siebel Analytics applications repository (SiebelAnalytics.rpd). It explains how to use the Merge Repository feature of the Administration Tool to merge your repository customizations of prior releases with a new version received from Siebel Systems. The Merge Repository feature can also be used by customers to upgrade their custom repositories.

The merge process involves three versions of a Siebel 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 Analytics.
- **Modified repository.** The repository that contains the customizations you made to the original repository.

- **Current repository.** The repository that is installed with this version and is currently opened as the main repository.

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 whether you want to merge your customizations with the current repository.

In this section, the following file names are used:

- Original file = SiebelAnalytics.Prev.rpd
- Current file = SiebelAnalytics.Current.rpd
- Modified file = SiebelAnalytics.Custom.rpd

To merge versions of Siebel Analytics Repository

- 1** In the Administration Tool, open 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 SiebelAnalytics.Prev.rpd.
- 4** Type the password and click OK.

The Merge Repositories dialog box displays the changes between SiebelAnalytics.Prev.rpd and SiebelAnalytics.Current.rpd.

- 5** In the Modified Repository field, select SiebelAnalytics.Custom.rpd.

In the Merge Repositories dialog box, the names of the original and modified repositories appear at the top. The Select buttons to the right allow you to select repositories.

The text box below the repository names allows you to select one of the following options:

- Changes from original repository to modified repository.
- Changes from original repository to current repository.

- How to merge contents of modified repository into current repository.

The grid below the text box dynamically changes based on the option you select. If you select the option to merge repository contents, the grid below the text box displays the following information:

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	Allows you to select a decision about what action you want to take regarding the repository changes.

Also, if you select the option to merge repository contents, the read-only text box below the grid displays a description of the object you select in the grid. Below the read-only text box are three panes. The left pane represents the original repository; the middle pane, the modified repository; and the right pane, the current repository. When you select an object in the grid, the object and its related repository objects are displayed in a hierarchical tree format in the appropriate pane.

- 6 Select the option How to merge SiebelAnalytics.Custom.rpd into SiebelAnalytics.Current.rpd from the drop-down list.

The grid displays the repository changes. The Description column states the change between the original repository and the current repository, and between the original repository and the modified repository. The Decision column provides options for the action you can take regarding the change. The read-only text box below the grid provides detailed information about the repository change for the object you select in the grid.

- 7 In the Decision field, select the action you want to take regarding the repository change, or accept the default action.

You can use the Find and Find Again buttons to locate specific objects by name.

- 8 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.

- 9 Click Merge.

A message appears to let you know that the merge is complete.

- 10 Save the current repository.

Creating an Analytics Multiuser Development Environment

Multiple Analytics developers can work on repository objects from the same repository during group development of Analytics applications.

NOTE: To perform the tasks in this section, administrators must understand the metadata creation process.

This section contains the following topics:

- [Business Scenario for Analytics Multiuser Development](#)
- [Setting up an Analytics Multiuser Development Environment](#)

Business Scenario for Analytics Multiuser Development

After completing an implementation of data warehousing at a company, the administrator wants to deploy Analytics to other functional areas of the company. Different experts in each functional areas will need to develop metadata to include in this repository. This requires that multiple people work concurrently on subsets of the metadata and merge these subsets back into a source repository without their work conflicting with other developers.

In some organizations, a single developer might manage all development. For simplicity and performance, the developer might want to maintain metadata code in smaller chunks instead of in a large repository.

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:

- Identify or create a shared network directory that will be dedicated to multiuser development.
- Copy the your repository file to that directory where it will be used as your source repository for multiuser development.

NOTE: In this section, we use the phrase source repository to refer to this copy of a repository in the multiuser development directory.

- Create the projects that your developers need in the multiuser source repository.

Setting Up the Shared Network Directory

The administrator needs to identify or create a shared network directory 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 source 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 shared network directory that is dedicated to multiuser development. If not set up and used as specified, repository files can be unintentionally overwritten and repository data can be lost.

- Make a copy of the appropriate repository files and paste them in the directory that you have dedicated to multiuser development. This copy (source repository for multiuser development) will be downloaded by the developers who will make changes and then merge these changes back into the source repository.

Creating Projects for a Multiuser Development Environment

After the Administrator copies a repository to the shared network directory, the Administrator needs to create distinct projects in the source repository in the shared network directory.

A project is a defined subset of metadata in the source repository. 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 source repository. When defined this way, projects 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 source 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 a source repository that you copied to the multiuser development directory, 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.
- 5** 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.

- In the Project dialog box, select a presentation catalog and then click Add.

The Administration Tool automatically adds all the logical fact tables.

- 6 To remove any fact table from the project, in the right pane, select the fact table and click Remove.
- 7 Add any Groups, Users, Variables, or Initialization Blocks needed for the project.
- 8 Type a name for the project.
- 9 Click OK.

After you create all projects, 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 source repository. This must be the multiuser development directory created by the administrator.

During checkout and checkin, a copy of the source 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 source repository.

To make changes in a multiuser development environment, perform the following tasks:

- [Setting Up a Developer's Machine for Analytics Multiuser Development](#)
- [Changing Metadata in an Analytics Multiuser Development Environment on page 195](#)
- [Merging Metadata Changes Into the Source Repository on page 198](#)

Setting Up a Developer's Machine for Analytics Multiuser Development

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 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.

Changing Metadata in an Analytics Multiuser Development Environment

After setting up a pointer to the source repository, a developer can check out projects, change metadata, and test the metadata.

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 source repository and writes an entry in the log file for the source repository file ([source repository].rpd.log). The entries in this log file record the time and date when 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 source repository file will be automatically deleted.

- In the multiuser directory on the network, the Administration Tool writes an entry in a log file ([*source 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 originalDevelopment1.rpd and Development1.rpd based
on project(s) Supplier Sales KW2 Login 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 uses this copy for metadata changes.

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.

Additionally, 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 source repository is automatically deleted.

CAUTION: The Administration Tool does not place a lock on the projects in the source repository. Therefore, nothing 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 directory on the shared network drive.

To check out projects

- 1 From the Administration Tool menu, choose File > Multi-User Checkout.
- 2 In the Select repository to extract subset from dialog box, select the source repository and then click Open.

CAUTION: If a repository file with that name already exists in this location, a message appears asking if you want to overwrite it. If you click Yes, the file in your local \SiebelAnalytics\Repository directory will be immediately overwritten and after the new repository is saved, the source repository file will be automatically deleted.

- 3 In the Extract from [*name of repository*] 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, and then click OK.
- 5 In the New Repository dialog box, type a name for the new repository and then click Save.

After you specify the name of the repository containing the project you want to change, the Administration Tool opens this repository in offline mode.

Changing and Testing Metadata

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. For more information about testing the metadata, see [“Process of Completing the Setup for a Repository File” on page 175](#).

NOTE: DSNs specified in the metadata need to exist on the developer's workstation.

Merging Metadata Changes Into the Source Repository

After changing and testing the metadata, the developer initiates a merge of the metadata changes into the source repository. The Administration Tool automatically copies the current version of the source repository from the shared multiuser directory to the \\SiebelAnalytics\Repository directory on the developer's machine and updates the log files in the local and multiuser development directories. This is necessary because the source repository in the shared multiuser directory might have changed after the developer checked out the projects.

The merge process is a three-way merge involving the following files:

- **Source repository.** This may have been modified by other developers before this merge.
- **Modified local (subset) repository.** Contains the projects modified by the developer.
- **Original copy 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.

Only one developer at a time can merge metadata from a local repository into the source repository. When the check-in process begins, the Administration Tool determines if the source repository is currently locked. If not, it locks the source 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 Checkin option on the File menu will be unavailable until the in-process merge is complete.

Multiuser Development Merge Logic

The multiuser development checkin process uses the same technology as the standard repository merge but with several important differences. For more information about the standard repository merge, see [“Merging Siebel Analytics Repositories” on page 188](#). The following is a list of the way in which the multiuser development merge is different from the standard repository merge:

- Inserts are applied automatically. Because a subset of the source repository is being used as the original repository, most objects in the source 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 merge metadata changes into the source repository

- 1** From the Administration Tool menu, choose File > Multi-User Checkin.
- 2** In the Merge repositories dialog box, verify that the Original repository and Modified repository file names are correct.

The center pane contains conflict resolution information.

- 3** In the center pane, make the choices required to resolve any conflicts.
- 4** Click Merge and save the repository.

Completing Setup and Managing Repository Files

Making Changes in an Analytics Multiuser Development Environment

Setting Up Disconnected Analytics

9

Siebel Disconnected Analytics allows mobile users to access data and reports when not connected to a network. For more information about using Disconnected Analytics on a mobile user's machine, see *Siebel Disconnected Analytics Online Help*.

Siebel Analytics provides the following disconnected 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 Analytics Web. For more information about Briefing Books, see *Siebel Analytics Web Administration Guide*.
- Managed Disconnected Analytics is centrally administered and managed. It provides most analytical functionality that is available in the network-based Siebel 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 dashboards on Siebel Analytics Web.

NOTE: In this guide, Disconnected Analytics refers to managed Disconnected Analytics. The disconnected database, disconnected repository, and other disconnected components are interchangeably referred to as mobile, local, or disconnected. When Siebel Analytics is used to describe components, it refers to components of the network-based Siebel Analytics application. For example, Siebel Analytics Server refers to the network-based Siebel Analytics Server.

Disconnected Analytics mobile users typically perform the following tasks:

- Download data from an Siebel Analytics database to a SQL Anywhere database on a laptop.

NOTE: SQL Anywhere is a mobile relational database that is installed with Siebel Analytics.

- Use interactive analytical dashboards.
- If mobile users have authorization to access Siebel Answers, they can create and modify their own sourcing reports that determine what data is downloaded. For more information about sourcing reports, see [“Defining Sourcing Reports for Disconnected Analytics”](#) on page 216.

This section contains the following topics:

- [“Disconnected Analytics Architecture \(Server and Laptop\)”](#) on page 203
- [“Installing Siebel Disconnected Analytics”](#) on page 205
- [“Process of Creating and Deploying Disconnected Analytics Applications”](#) on page 206
- [“Creating and Testing the SQL Anywhere Database Tables”](#) on page 206
- [“Creating the Disconnected Analytics Repository”](#) on page 210
- [“Creating the Disconnected Web Catalog”](#) on page 215
- [“Defining Sourcing Reports for Disconnected Analytics”](#) on page 216
- [“Defining Data Sets for Disconnected Analytics”](#) on page 218
- [“Creating the Disconnected Application Configuration File”](#) on page 219
- [“Preparing Disconnected Analytics Applications for Deployment”](#) on page 226
- [“Testing and Deploying Siebel Disconnected Analytics”](#) on page 229

Disconnected Analytics Architecture (Server and Laptop)

Figure 14 on page 204 shows the Siebel Analytics Server and disconnected laptop components of Siebel Analytics Server and Siebel Analytics Web. This figure also shows that sourcing reports populate the local tables from the network-based Siebel Analytics repository. You will use two different analytics platforms, each with a set of components. One platform is on the network-based server and one is on the mobile machine. The difference between the Siebel Analytics Server and disconnected server components is that the disconnected components are scaled and configured for single use.

Data is downloaded to a mobile machine using sourcing reports and the Siebel Disconnected Analytics Application Manager. The administrator or mobile user creates sourcing reports that download data for a specific application. Sourcing reports are created in Siebel Answers, stored in the Siebel Analytics Web Catalog, and executed from the application configuration file that is run by the Disconnected Analytics Application Manager.

The Disconnected Analytics Application Manager is installed on the laptop at the same time as Siebel Disconnected Analytics. It is a small utility that allows a mobile user to download tables and application data to a laptop database. It does not upload data from the laptop to the Siebel Analytics Server. For instructions about how to use Disconnected Analytics Application Manager, see *Siebel Disconnected Analytics Online Help*.

CAUTION: The Disconnected Analytics Application Manager is associated with the DAD and SDC file extensions. You should not edit a DAD or SDC file.

Setting Up Disconnected Analytics

Disconnected Analytics Architecture (Server and Laptop)

When a mobile user downloads application data, Disconnected Analytics Application Manager executes the sourcing reports on the Siebel Analytics Web Server. Siebel Analytics Web runs the sourcing reports and compresses the data for downloading to the laptop. The download operation and the subsequent load into the local SQL Anywhere database is performed by Disconnected Analytics Application Manager.

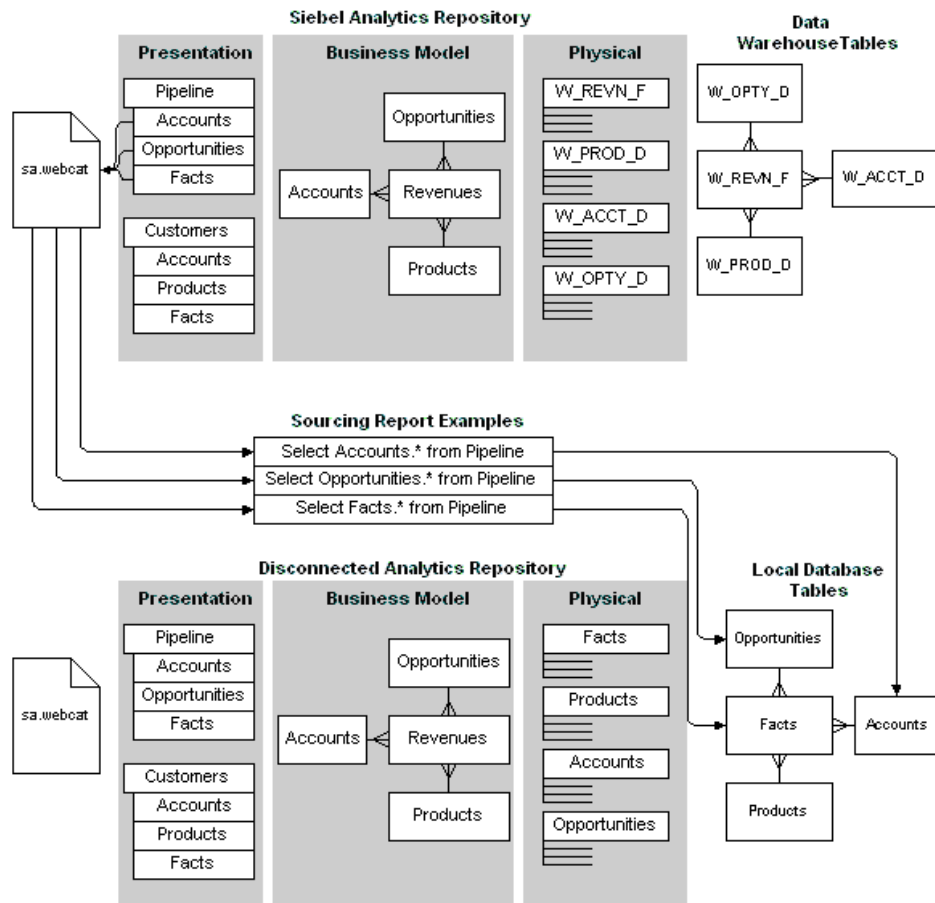


Figure 14. Disconnected Analytics Sourcing Reports Populate the Local Tables

Installing Siebel Disconnected Analytics

Disconnected Analytics is not installed at the same time as network-based Siebel Analytics. You cannot install both products on the same machine. Installing Disconnected Analytics requires the following separate installations:

- **Standard Siebel Analytics installation.** You must install Siebel Analytics on a network-based server and prepare for the disconnected installation by completing the tasks in [“Process of Creating and Deploying Disconnected Analytics Applications.”](#)
- **Disconnected client installation.** After completing the setup tasks, you need to install the disconnected client on each mobile machine that needs to access a disconnected application.

If you purchased Disconnected 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 Disconnected Client.

For installation instructions, see *Siebel Analytics Installation and Configuration Guide*.

NOTE: Several components for Disconnected Analytics have been tailored for the Siebel Pharma product. These components are installed during the Siebel Analytics installation in the \SiebelAnalyticsData\Disconnected\Pharma directory. For more information, see [“Pharma Analytics Administration Reference”](#) on page 481.

For more information about sourcing reports and data sets, see [“Pharma Analytics Administration Reference”](#) on page 481.

Process of Creating and Deploying Disconnected Analytics Applications

Each Disconnected Analytics application contains several components such as a repository and Web Catalog that are slightly different from their counterparts on the network-based Analytics Server. Because Siebel Disconnected Analytics provides only local platform functionality and a few utilities, the administrator needs to create each disconnected application by building a local database, a customized local repository, a local Web Catalog, sourcing reports to download data, and DDLs (data definition language). After you create an application, you can test, and then deploy it.

Perform the following tasks to create and deploy a disconnected application:

- [Creating and Testing the SQL Anywhere Database Tables on page 206](#)
- [Creating the Disconnected Analytics Repository on page 210](#)
- [Creating the Disconnected Web Catalog on page 215](#)
- [Defining Sourcing Reports for Disconnected Analytics on page 216](#)
- [Defining Data Sets for Disconnected Analytics on page 218](#)
- [Creating the Disconnected Application Configuration File on page 219](#)
- [Preparing Disconnected Analytics Applications for Deployment on page 226](#)
- [Testing and Deploying Siebel Disconnected Analytics on page 229](#)

Creating and Testing the SQL Anywhere Database Tables

Before you can build or modify sourcing reports, you need to set up a SQL Anywhere database with the appropriate tables. Each disconnected application such as sales or service requests has local database tables. When making design decisions, perform the following tasks:

- [Design the Table Schema for the Disconnected Analytics Application](#)
- [Write the DDLs to Create the Local Database Tables on page 207](#)

- [Creating and Testing Local Database Tables](#)

Design the Table Schema for the Disconnected Analytics Application

Identify the tables and their columns (table schema) that will form the data for the disconnected application. This is a critical step because the tables, their definitions, and structure determine the structure of the disconnected repository. Therefore, it is important to allocate enough time to design these tables correctly.

The administrator should decide what the disconnected database tables will be named. The names of the disconnected tables are not required to be the same as the names of the Siebel Analytics data warehouse tables. However, using the same table names in the disconnected database makes it easier to build the Physical layer of the disconnected repository and perform debugging tasks while developing the disconnected application.

The following are the primary tasks to perform in making your design decisions.

- Decide what tables and columns you need in a disconnected application. This will be your table schema.
- Using the Siebel Analytics Presentation layer, find the tables and columns that will give you the data that you need.

You will also use this design information to perform the following tasks:

- Determine the structure of the disconnected Physical layer. For more information, see [“Create the Physical Layer of the Disconnected Analytics Repository” on page 211](#).
- Design sourcing reports that extract data from the Siebel Analytics database. For more information, see [“Defining Sourcing Reports for Disconnected Analytics” on page 216](#).

Write the DDLs to Create the Local Database Tables

Data is stored in the local database in a relational format. Administrators write the DDLs (data definition language) that create the tables in the local database. The Disconnected Analytics Application Manager running on the laptop uses these DDLs to create the tables and set up the appropriate indexes.

On the mobile user's laptop, the Disconnected Analytics Server uses the local database to store and query data. If an application has multiple tables, then you do not need to setup the joins in the database. The joins can be setup in the local repository.

The following two examples show sample DDLs. One is an example of a DDL that creates a table with the appropriate columns and one is an example of a DDL that creates an index for a table.

Example of a DDL File That Creates a Table

The following is an example of a file called SRFlat.sql that creates a table and several columns.

```
drop table SRFlat;
create table SRFlat
    (OpenDate           Date,
    OpenMonth           varchar(20),
    OpenYear            varchar(20),
    AccountLocation     varchar(50),
    AccountOrganization varchar(50),
    AccountType         varchar(50),
    AccountName         varchar(50),
    Product             varchar(50),
    ProductLine         varchar(50),
    PartNumber          varchar(50),
    AssetNumber         varchar(50),
    NumberofSRs         int,
    NumberofClosedSRs  int,
    NumberofOpenSRs    int,
    NumberofPendingSRs int,
    NumberofCancelledSRs int);
```


Example of a DDL That Creates an Index for a Table

The following is an example of the contents of a file that creates an index for a table called SRFlatIndex.sql:

```
DROP index SRFlatindex;  
create index SRFlatindex on SRFLat (AccountOrganization,Product);
```

NOTE: Data types supported for the database tables are varchar, char, int, decimal, date, timestamp, and numeric(p,s).

Creating and Testing Local Database Tables

Pointers to the DDLs are in the application configuration file and the database tables are created or overwritten when the application configuration file is executed by the Disconnected Analytics Application Manager.

NOTE: Disconnected Analytics Application Manager bulk loads the data into the local database. It does not check for duplicates and always appends data. It is the administrator's responsibility to verify data integrity.

It is recommended that you use SQL Anywhere to test creation of the disconnected tables (your DDLs). You can choose to use another relational database such as MS SQL Server during development. However, before you can deploy a disconnected repository created in another relational database to mobile users, you must change the database connection pool values in the repository to the required disconnected SQL Anywhere values. For information about SQL Anywhere connection pool values, see [“Create the Physical Layer of the Disconnected Analytics Repository” on page 211](#).

Creating and Testing SQL Anywhere Database Tables

If you decided to use SQL Anywhere during development, Siebel Systems provides the Sybase Interactive SQL utility (dbisqlc.exe) to help you perform developmental tasks. 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 contact Sybase.

Dbisqlc.exe allows you to create and test SQL Anywhere database tables. The following list contains general instructions:

- Use the SiebelAnalytics.db database (shipped with Siebel Analytics) to create tables.

NOTE: The user name is dba and the password is sql.

- Use dbisqlc.exe to test the DDLs for disconnected database users.
- Use dbisqlc.exe to test the DDLs for standalone disconnected users.

Creating the Disconnected Analytics Repository

Create the disconnected repository using the table structure defined in [“Design the Table Schema for the Disconnected Analytics Application”](#) on page 207.

An administrator creates a disconnected repository for each application such as sales, based on your business requirements. The disconnected repository is conceptually a pass-through repository because it is derived from the Siebel Analytics repository. The logical data extracted from the Siebel Analytics Server is already aggregated and transformed from the Siebel Analytics physical data.

NOTE: It is recommended that you do not make the disconnected repository complicated or large (more than 20 MB). This can affect the performance of the client environment.

For development purposes, the disconnected repository should be set up and tested on a laptop that mirrors the mobile user's environment as closely as possible.

To create a disconnected repository, perform the following tasks:

- [Create the Physical Layer of the Disconnected Analytics Repository on page 211](#)
- [Create the Business Model and Mapping Layer of the Disconnected Repository on page 213](#)
- [Create the Presentation Layer of the Disconnected Repository on page 214](#)

- [Testing the Disconnected Repository on page 214](#)

Create the Physical Layer of the Disconnected Analytics Repository

The Physical layer of the Disconnected Analytics repository should be configured to run against a SQL Anywhere database. This allows the Siebel Analytics Server to generate the appropriate SQL statements for downloading data.

You can import the Physical layer schema database using the import feature in the Administration Tool. If you set up the tables in a local SQL Anywhere database, you can import these tables into your disconnected repository. For detailed instructions about how to create the Physical layer, including importing the physical schema, see [Chapter 5, “Creating and Administering the Physical Layer in a Repository”](#) and [“Importing a Physical Schema” on page 83](#).

The following list discusses unique properties for the disconnected database and disconnected connection pool:

- The database data source definition must be SQL Anywhere as shown in [Figure 15](#).

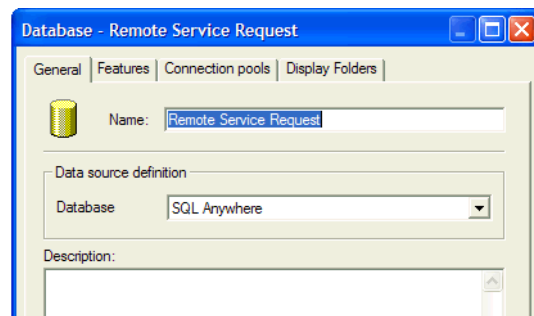


Figure 15. Disconnected Data Source Definition

- [Figure 16](#) shows the connection pool parameters in the Connection Pool dialog box.

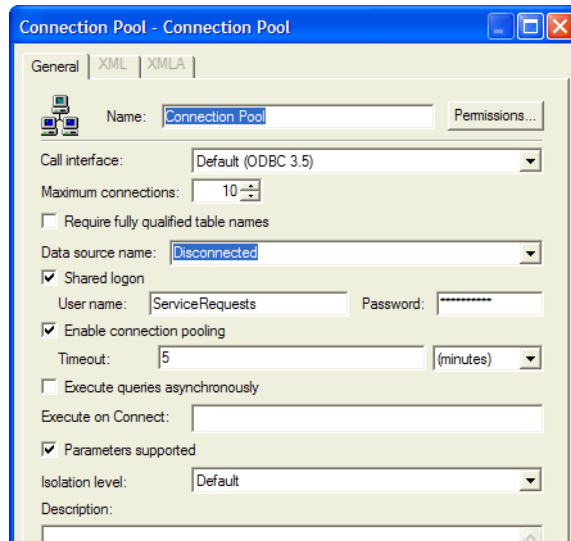


Figure 16. Disconnected Connection Pool Properties

- The connection pool data source name must be Disconnected because client laptops are configured during the Disconnected Analytics installation to use this data source name. For an example of connection pool parameters, see [Figure 16 on page 212](#).
- The user name and password of the connection pool must be the same as the name of the application. This application name is specified in the application configuration file and the user name and password must match it. For more information, see [“Creating the Application Configuration File for Disconnected Analytics” on page 220](#).
- The mobile user does not need to know the user name and password for the local 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 disconnected applications.

- For authentication purposes, you can add a standard user name in the disconnected repository for all mobile users to connect to their local Disconnected Analytics application.
- Data is always personalized, because it is extracted from the Siebel Analytics Server using each user's name and password.
- If your company uses 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 326](#).
- Column names in a database table cannot contain special characters such as #. Change fact table column names to contain alphanumeric characters only.

Create the Business Model and Mapping Layer of the Disconnected Repository

After you build the disconnected Physical layer, you can build the Business Model and Mapping layer. The Business Model and Mapping layer for the disconnected repository is similar to the Presentation layer of the network-based Siebel Analytics repository. The following list includes information about creating the Business Model and Mapping layer for a disconnected repository:

- The source tables for disconnected logical fact tables will be different from the fact tables in the network-based Analytics repository. The source tables in the disconnected repository point to the disconnected Physical layer tables. The source tables in the network-based Siebel Analytics repository point to data warehouse tables.

You can copy the logical layer objects from the Siebel Analytics repository, paste them in the disconnected Business Model and Mapping layer, and then change the source tables. The difference between the Siebel Analytics source tables and the disconnected source tables is the presence of aggregation tables in the Siebel Analytics Physical layer.

- Dimensional hierarchies should be the same in the Siebel Analytics and disconnected repositories to achieve the same drill-down behavior.

- Administrators can choose to remove some of the lowest level granular data to make the data sets smaller. For more information about data sets, see [“Defining Data Sets for Disconnected Analytics”](#) on page 218.

For more information about creating the Business Model and Mapping layer of the Siebel Analytics repository, see [Chapter 6, “Creating and Administering the Business Model and Mapping Layer in a Repository.”](#)

Create the Presentation Layer of the Disconnected Repository

To create the disconnected Presentation layer, administrators should use the same names as in the network-based Siebel Analytics repository. Typically, the names would be the same because you used the network-based Siebel Analytics Presentation layer to create the disconnected Physical layer. For more information about creating the Presentation layer of the Siebel Analytics repository, see [Chapter 7, “Creating and Maintaining the Presentation Layer in a Repository.”](#)

Testing the Disconnected Repository

The following is a list of high-level steps that you might use to help you test your disconnected repository:

- Identify a machine to test the disconnected repository. This machine can either be an network server or a laptop.
- Install the disconnected client on this machine. The installation creates a disconnected version of the Siebel Analytics Server that is scaled for single use.
- Create the disconnected database on this machine by using dbisqlc.exe to manually create the tables.
- Deploy the disconnected repository on the Siebel Analytics Server by editing the nqsconfig.ini file to point to the disconnected repository. For information on the NQSConfig.INI file parameters, see *Siebel Analytics Installation and Configuration Guide*.
- Use the disconnected client to connect to the server (the default DSN name is Analytics Web) and view the schema.

CAUTION: Do not try to use Siebel Analytics Web yet.

- Run a sample logical SQL query 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 Disconnected Web Catalog

The Disconnected Web Catalog can submit requests only to the disconnected repository. If the subject area, table, and column names are identical in the disconnected and network-based Siebel Analytics repositories, the Disconnected Web Catalog can be a duplicate of the network-based Siebel Analytics Web Catalog or a copy scaled for single use. When Siebel Disconnected Analytics reads the catalog, it optimizes it for a single disconnected use. For example, the mobile user's personal folders are maintained but changes made to shared folders are overwritten by the contents of the Siebel Analytics Web Catalog.

NOTE: Web Catalog changes made by mobile users are not uploaded to the Siebel Analytics Web Catalog.

To create a disconnected Web Catalog, use the following guidelines:

- After you create the disconnected repository, use it to test and modify the disconnected dashboards using Siebel Analytics Web. Disconnected dashboards are saved in the disconnected Web Catalog.
- It is recommended that the Presentation layer object names in the disconnected repository should match Presentation layer object names in the Siebel Analytics repository. If they do not, you must build the disconnected Web Catalog based on the disconnected repository.
- To create the disconnected Web Catalog, use one of the following methods:

- Use the network version of the Web Catalog (typical method). If you maintained the network version of the Presentation layer names and structure in the disconnected repository, the Web Catalog integrates more smoothly with the disconnected environment. To use this method, make a copy of your network Web Catalog, remove unnecessary items to make it more relevant for the disconnected user, and copy it to the application directory.

NOTE: You might want to rename the disconnected Web Catalog file to differentiate it from the network-based Web Catalog.

- Use a new Web Catalog. If you decide not to use the network-based Siebel Analytics Web Catalog, then start with a blank Web Catalog. To use this method, open Siebel Analytics Web. When it does not find a Web Catalog, it creates one by default. You can use this empty Web Catalog to start your disconnected application. For more information, see *Siebel Analytics Web Administration Guide*.

Defining Sourcing Reports for Disconnected Analytics

The sourcing report is an analytics query, created in Siebel Answers and stored in the Siebel Analytics Web Catalog on the network. For instructions about how to create reports in Siebel Answers, see *Analytics Web Online Help*. Your sourcing reports design is linked to the local database table schema and should be based on how your business is organized. For information about creating the table schema, see [“Design the Table Schema for the Disconnected Analytics Application” on page 207](#)

The order of the columns in a sourcing report has to match the order of columns in the table that is populated by that sourcing report. Sourcing reports can contain global filters (apply to every user) that you add to the source query. Additionally, users can choose their own filter parameters on certain columns. To allow users to choose their filters, you need to add the appropriate columns during application configuration. For more information, see [“Creating the Disconnected Application Configuration File” on page 219](#).

When a mobile user requests a download (synchronization), Disconnected Analytics Application Manager sends the request to the Siebel Analytics Web server. Siebel Analytics Web uses this sourcing report to query the Analytics Server, obtain data, and package it into a data set. Siebel Analytics Web returns the application data to the mobile user's disconnected database and populates the mobile database tables.

NOTE: At this time, synchronization is a one-way process only. You can download data from a network-based Siebel Analytics database but you cannot upload data from a laptop to a network-based Siebel Analytics database.

Mobile users can also request sourcing reports to be run in real time (Online) if the environment is configured to allow this. In Online processing, data sets are customized for each user based on the user's permissions and responsibilities, and then the data on each mobile user's local database is filtered for each user.

Figure 17 is a high-level illustration of how the local database tables are populated from the Web Catalog of the network-based Siebel Analytics repository.

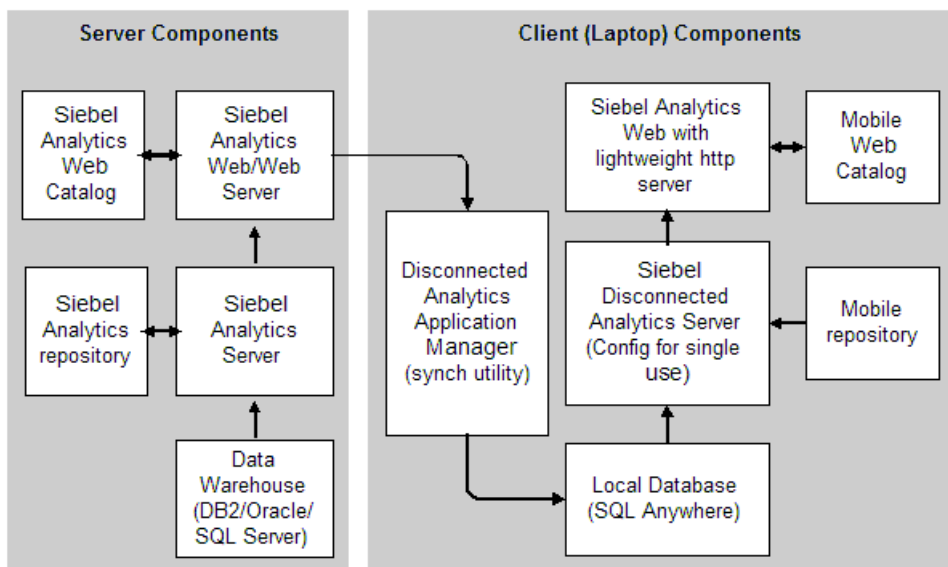


Figure 17. Sourcing Reports Populate the Local Database Tables

Defining Data Sets for Disconnected Analytics

Sourcing reports populate local database tables using a data delivery mechanism called a data set. A data set is a logical entity that allows you to group (organize) tables for efficient deployment. Data sets are a way to deliver data to mobile users and do not affect the local application database structure. For example, facts change more often than dimensions and data from fact tables would typically be provided to mobile users more often. Therefore, you might decide to assign facts and dimensions to separate data sets so that these more frequent updates would be smaller. If data from external sources such as syndicated data (obtained from data syndication agencies) needs to be sent to users, this data could be put into a third data set.

You specify data sets in an application configuration file. An application configuration file describes one or more data sets and each data set references one or more tables. Tables should be grouped in a data set based on the way you organize your business. Each table is associated with a sourcing report. Siebel Analytics Web uses this sourcing report to query the Analytics Server, obtain data, and package it into a data set for delivery.

CAUTION: It is the administrator's responsibility to ensure 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.

Creating the Disconnected Application Configuration File

The disconnected application configuration file names all the Disconnected Analytics elements required to use Siebel Disconnected Analytics. This file does not contain any data. The Disconnected Analytics Application Manager gets all its information from this application configuration file.

The application configuration file is an XML file that must contain references to the following application elements:

- Name of the application
- Pointer to the sourcing report in the Siebel Analytics Web Catalog
- Pointer to the disconnected repository and disconnected Web Catalog
- Pointer to the DDLs that create the tables and index
- Specification of data sets and their table groupings

You have to create a unique application configuration file for each Disconnected Analytics application. For information about where the application configuration file is stored, see [“Creating the Disconnected Application Directories” on page 227](#).

NOTE: If you purchased Disconnected Analytics to use with your Siebel Pharma application, the application configuration file is preconfigured.

Creating the Application Configuration File for Disconnected Analytics

This section contains two examples for you to use when creating your application configuration files in XML format. One is an example of a simple application configuration file structure. In this example, there is one data set for fact table data and one data set for dimension table data.

A second example is of a completed application configuration file.

Application Configuration File Structure Example

The general structure of the application configuration file is in the following example:

```
Begin Configuration File

Application: My Sales Application

Component Webcatalog: SalesApp.webcat

Component repository: SalesApp.rpd

Component postdbschema: SalesApp.sql

Data

  Data set Name: Sales Facts

    Table: Facts1

      Component sourcing report: facts1 SR

    Table: Facts2

      Component sourcing report: facts2 SR

  Data set Name: Sales Dimensions

    Table: ComDims

      Component sourcing report: facts3 SR

End Configuration File
```

Application Configuration File Example

The following example of an application configuration file is provided with the installation:

```
<remotecfg>

  <application name="ServiceRequests" displayname="Service Requests"
  dir="app">

    <webcatalog name="ServiceRequest.webcat" />

    <repository name="ServiceRequest.rpd" />

    <postdbschema name="SRFlatpostdb.sql" />

  </application>

  <data dir="data" catalogfolder="/shared/Disconnected">

    <data set name=="Service Requests" rank="1" validitytype="date-
    based" expiry="2003-12-12" syncdefaultfirsttime="true"
    syncdefaultsubsequenttimes="true" subjectarea="ServiceRequest">

      <table name="SRFlat">

        <sourcingreport name="ServiceRequest" file="SR_Flat.csv" />

        <tablesql name="SRFlat.sql" />

        <indexsql name="SRFlatindex.sql" />

      </table>

    </data set>

  </data>

</remotecfg>
```

XML Tags for Disconnected Analytics Application Configuration Files

Table 22 contains a list of the XML tags used most frequently in application configuration files. All tags and properties must be in lowercase.

Table 22. XML Tags for Disconnected Analytics Application Configuration File

File Tag	Properties	Values or Description	Type	Optional
< application > < / >	name	Name of the application. This is not exposed to the mobile user	String	No
	displayname	Display name of the application. This is the descriptive name that is exposed to the mobile user through the Disconnected Analytics Application Manager	String	No
	dir	Name of the folder under which the application files are stored. Files include the repository, the Web Catalog and DDL files	String	No
< webcatalog > < / >	name	Disconnected Web Catalog name. This Web Catalog is used by the disconnected Analytics Web to store dashboards and reports.	String	No
< repository > < / >	name	Disconnected Analytics Server reads the metadata from this repository	String	No
< postdbschema > < / >	name	Any SQL that will be executed for cleanup and maintenance.	String	No
< data dir > < / >	name	Name of the directory that is used by the Analytics Server to place the data for each user.	String	No

Table 22. XML Tags for Disconnected Analytics Application Configuration File

File Tag	Properties	Values or Description	Type	Optional
	catalogfolder	Full path of the Analytics Web Catalog folder that contains the sourcing reports	String	No
< dataset > < / >	name	Data Set name	String	No
	rank	Defines the order in which the data set is loaded	Number	No
	validitytype	Required for all data sets. The following is a list of values: <ul style="list-style-type: none"> ■ date-based ■ rolling-period 	Keyword	No
	start	Gives start date in yyyy-mm-dd format only. Only used if validitytype is date-based. Cannot be set to blank.	Date	Yes
	expiry	End date of the data set (Only used if validitytype is date-based)	Date	Yes
	period	Number of period units (Only used if validitytype is "rolling-period")	Number	Yes
	periodunit	The unit in which the period is defined. Must be one of: "year", "quarter", "month", "week" and "day"	Keywords	Yes
	dependson	Name of another data set on which this data set depends	String	Yes
	syncdefaultfirsttime	Synchronizes the data set the first time by default. Values must be True or False	Boolean keyword	Yes

Table 22. XML Tags for Disconnected Analytics Application Configuration File

File Tag	Properties	Values or Description	Type	Optional
	syncdefaults ubsequenttimes	Synchronizes the data set every single time. Values must be True or False.	Boolean keyword	Yes
	syncmode	Available value is online. Users can get data online while connected.	Keyword	Yes
	forcesync	"yes" causes the data set to be synchronized regardless of whether the user chooses to do so	Boolean keyword	Yes
	subjectarea	Required for every data set	String	No
< table > < / >	name	Table name in the SQL Anywhere database	String	No
< sourcing report > < / >	name	Name of the analytical query that generates the data	String	No
	file	This is the file name used for storing data on disk for the given sourcing report. In case the sourcing report is left NULL, then file can be pointed to an external data file on the disk	String	No
< filterables > < / >		A child tag of < sourcing report > Specifies a list of filterable columns for each table for the stated sourcing report.	String	Yes
< formula > < / >		Child tag of < filterables > . The column in the subject area in which to filter the data.	String	Yes

Table 22. XML Tags for Disconnected Analytics Application Configuration File

File Tag	Properties	Values or Description	Type	Optional
< tablesql > </ >	name	Name of the SQL file that contains the DDLs to create the tables in the database	String	No
< indexesql > </ >	name	Name of the SQL file that contains the DDL that sets up the indexes	String	No

Preparing Disconnected Analytics Applications for Deployment

After you create the application configuration file, you need to test the application elements and the deployment process. When these tests are complete, 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 Disconnected Analytics client is installed on each mobile user's laptop.

Perform the following tasks to install a disconnected application on a mobile user's laptop:

- [Creating the Disconnected Application Directories](#)
- [Creating a Silent Installation for Disconnected Application Deployment on page 227](#)
- [Testing Disconnected Analytics Applications on page 229](#)

Creating the Disconnected Application Directories

The administrator should create a separate directory named for each application to be deployed. The application directory must be created in the `\SiebelAnalyticsData\Disconnected\` directory so that each application appears on the disconnected application page for downloading. You need to set up the following directory structure:

`\SiebelAnalyticsData\Disconnected` (Contains a subdirectory for each application you deploy and a subdirectory for data.)

`\[application name]` (Contains the application files required for each disconnected application deployment. Must be an exact match with the application name specified in the application configuration file.)

`[application name].xml` (This is the application configuration file. It is stored in the root directory of each application directory on the network drive)

`\App_Dir\`

`[application name].rpd` (repository file)

`[application name].webcat` (Web Catalog file)

`[application name].sql` (create table file)

`[application name].sql` (optional postdb.sql file. that cleans up after download)

`\Data_Dir\` (Must be an exact match with the directory specified in the application configuration file.)

Creating a Silent Installation for Disconnected 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 *Siebel Disconnected 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.

NOTE: Incremental synchronization is not available at this time.

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 230](#).

Make sure that Siebel Disconnected Analytics (client components) has been installed on the mobile machine (typically a laptop) before deploying.

To build a response file for a silent disconnected 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 Disconnected Client is being installed.  
Please wait...
```

```
setup.exe -options [complete path to response file] -silent  
  
echo Siebel Analytics Disconnected Client was successfully  
installed.
```

You will test the response file when you test the silent installation.

Testing and Deploying Siebel Disconnected 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 Siebel Disconnected Analytics (client components) on each mobile user's machine. For an overview of the install process, see [“Installing Siebel Disconnected Analytics” on page 205](#). For more details, see *Siebel Analytics Installation and Configuration Guide*.

To test and install disconnected applications, perform the following tasks:

- [Testing Disconnected Analytics Applications](#)
- [Testing the Installation Processes on page 230](#)
- [Installing an Application Using Disconnected Analytics Application Manager on page 230](#)
- [Installing a Disconnected Analytics Application Silently on page 231](#)
- [Deleting Disconnected Analytics Applications on page 232](#)

Testing Disconnected 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 local disconnected application independently of the Disconnected Analytics Application Manager utility. Conducting a test in this way validates the disconnected 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 Disconnected Analytics application

- 1 Identify a machine on which you will test.

- 2 Install the Siebel Analytics Server components on this machine.

For installation instructions, see *Siebel Analytics Installation and Configuration Guide*.

- 3 Use the disconnected repository and Web Catalog on this machine, in place of Siebel Analytics repository and Web Catalog.

- 4 Setup 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 Siebel Analytics Server and Siebel 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 disconnected application using Disconnected Analytics Application Manager. This section contains general instructions. For details about installing and using Disconnected Analytics and the Disconnected Analytics Application Manager, see *Siebel Disconnected Analytics Online Help*.

To test downloading a disconnected application

- 1 Identify a mobile machine on which you will test.
- 2 Install the Disconnected Analytics client components on this machine.

For installation instructions, see *Siebel Analytics Installation and Configuration Guide* and [“Installing Siebel Disconnected Analytics” on page 205](#).

- 3** Using the mobile machine, Start Disconnected Analytics Application Manager by choosing Start > Programs > Siebel Analytics 7.x > Disconnected Analytics Application Manager.
- 4** In Disconnected Analytics Application Manager, download the application.
For detailed instructions, see *Siebel Disconnected Analytics Online Help*.

To test downloading data to a local database

- 1** Using the mobile machine, open Siebel Analytics on the network and click the Disconnected link in the upper right hand corner.
- 2** The disconnected applications that have been deployed appear.
- 3** Click Update Data.
- 4** In the File Download dialog box, click Open.
- 5** In 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 *Siebel Disconnected Analytics Online Help*.
- 2** Debug when errors occur.

Installing a Disconnected 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 Disconnected Analytics Applications

You might need to delete a disconnected application from the network-based server if you do not want anyone to download that application.

NOTE: For instructions about deleting disconnected applications from a laptop, see *Siebel Disconnected Analytics Online Help*.

To delete a disconnected application

- 1** Delete the application directory and files from the \SiebelAnalyticsData\Disconnected\ directory.
- 2** Login to Siebel Analytics and navigate to Siebel Answers.
- 3** In Siebel Answers, delete sourcing reports from Siebel Analytics Web Catalog.

For more information about Siebel Answers, see *Siebel Analytics Web Administration Guide*.

Administration Tool Utilities and Expression Builder **10**

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.

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 234](#)
- [Synchronize Aliases on page 234](#)
- [Replace Wizard on page 235](#)
- [Copy Business Model with Presentation Catalog on page 236](#)
- [Siebel Analytics Event Tables on page 236](#)
- [Execute UDML on page 237](#)
- [Externalize Strings on page 237](#)
- [Rename Wizard on page 238](#)
- [Update Physical Layer Wizard on page 239](#)
- [Generating Documentation of Repository Mappings on page 241](#)

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** Click Execute.
- 3** Choose Time Series 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** Open the Synchronize Aliases utility by selecting Tools > Utilities > Synchronize Aliases from the Administration Tool toolbar.
- 2** 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.

- 3** (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.

- 4 (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.

- 5 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.

- 6 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.

- 7 To synchronize an individual table, click the original table and its alias to select them, then click the Synchronize button.

- 8 Click the Close button.

Replace Wizard

The Replace Wizard automates the process of replacing physical tables or columns in logical table sources by allowing the Siebel Analytics Server administrator to select the sources from those displayed. The wizard prompts the Analytics administrator to replace columns as well as tables.

To start the Replace Wizard

- From the Administration Tool toolbar, select Tools > Utilities > Replace Column or Table in Logical Sources, and then click Execute.

Copy Business Model with Presentation Catalog

This utility allows you to select a matching business model and presentation catalog to duplicate as well assign new names for the duplicates. This utility is similar to Duplicate Business Model with Catalog Folder (a right-click command) but gives you the option to assign new names.

NOTE: Aliases are not copied.

To copy a business model and its presentation catalog

- 1** Open the utility by selecting Tools > Utilities > Copy Business Model with Presentation Catalog from the Administration Tool toolbar.
- 2** Select the business model to copy.
- 3** Specify new names for the business model and its catalog in the New name boxes, and then click OK.

When the copy process is finished, the utility window closes automatically and the copied business model is shown in the Business Model and Mapping layer window.

Siebel Analytics Event Tables

This utility allows you to identify a table as a Siebel 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 302](#).

To start the Siebel Analytics Event Tables utility

- From the Administration Tool toolbar, select Tools > Utilities > Siebel Analytics Event Tables, and then click Execute.

Execute UDML

This utility allows you to create objects in the repository using UDML (Universal Data Modeling Language). This is a seldom-used utility designed for people who have knowledge of UDML syntax.

To start the Execute UDML utility

- From the Administration Tool toolbar, select Tools > Utilities > Execute UDML, and then click Execute.

Externalize Strings

This utility is primarily for use by translators to translate Presentation layer catalogs, tables, columns, and their descriptions. You can save these text strings to an external file with ANSI, Unicode, and UTF-8 coding options.

NOTE: Before using this utility, translators should consult with Siebel Systems.

To translate a string

- 1** Right-click on a Presentation Catalog in the Presentation layer and select the options Externalize Display Names and Externalize Descriptions.
- 2** Start the Externalize Strings utility by selecting Tools > Utilities > Externalize Strings from the toolbar, and then click Execute.
- 3** 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 Analytics Web.

- 4** Click Save.
- 5** In the Save As dialog box, choose a type of file and an Encoding value and click Save.

- 6 In the Externalized Strings dialog box, click Close to end the utility.

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.

To start the Rename Wizard

- From the Administration Tool toolbar, select Tools > 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. The wizard does not add any columns or tables that exist in the back-end database but not in the repository.

NOTE: This wizard is not available for repositories that are opened in read-only mode, because they are not available for updating.

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 information about connection pool settings, see [“Setting up Connection Pools” on page 91](#).

To update objects in the Physical layer

- 1 Start the Update Physical Layer wizard by selecting Tools > Utilities > Update Physical Layer from the Administration Tool toolbar, and then click Execute.

The databases in the Physical layer of the repository are listed in the left pane of the wizard.

- 2** Highlight the databases that you want to update in the left pane, and then click Add.

The databases move to the update list in the right pane.

To remove a database from the update list, highlight it and click Remove.

The workstation 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 datatype-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

- 3** To see more information about an object, click it.

The information window in the wizard displays additional information about the object, including information about where it is used in the repository.

- 4** Select the objects to update in the Physical layer of the repository, and then click Next.

The wizard alerts you if it needs to check any objects out.

The workstation running the Administration Tool connects to each back-end database. The wizard displays information about each update that will be made to objects in the Physical layer.

- 5** Review the information about each update.
- 6** If you decide that you do not want the wizard to update a particular object in the Physical layer, click the Back button and deselect the object.
- 7** Click Finish.

The wizard updates the objects in the Physical layer, and then closes automatically.

- 8 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 Administration Tool menu, select Tools > 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.

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 Siebel Analytics Server.

For information about using SQL with the expression builder, see [“SQL Syntax and Semantics” on page 417](#). For information about the SQL functions supported by the Siebel Analytics Server, see [“SQL Reference” on page 429](#).

This section includes the following topics:

- [“About the Expression Builder Dialog Boxes”](#)

- [“Expression Builder Toolbar” on page 244](#)
- [“Folders in the Selection Pane” on page 245](#)
- [“Example of Setting Up an Expression” on page 246](#)
- [“Navigating Within the Expression Builder” on page 247](#)
- [“Building an Expression” on page 248](#)

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

Figure 18 shows an example of an expression builder. The edit pane at the top of the dialog box allows you to edit the current expression. 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 middle pane is the Categories pane. It displays the available categories for the folder you select in the Selection pane. The right pane is the Building Blocks pane. It displays the individual building blocks for the category you select in the Category pane. The toolbar in the middle contains commonly used expression building blocks.

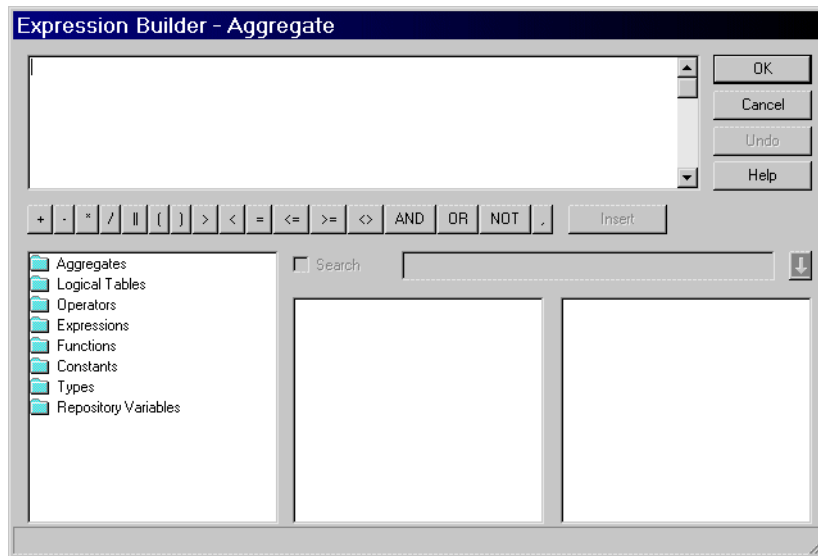


Figure 18. Example Expression Builder

Expression Builder Toolbar

The toolbar is located in the middle portion of the expression builder. [Table 23](#) describes each icon and its function in an expression.

Table 23. 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 19 shows the expression builder for a derived logical column, with a blank expression.

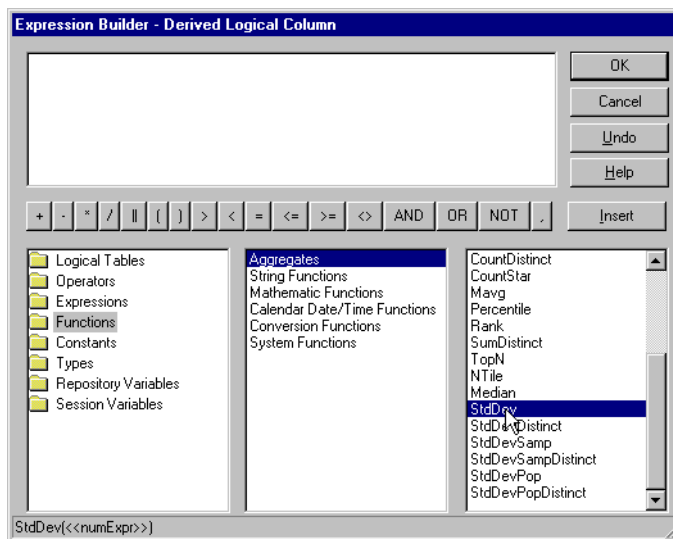


Figure 19. Expression Builder for Derived Logical Columns

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. Clicking once between the parentheses of the function darkens that area, marking it as the insertion point.

Double-clicking on the logical column pastes the logical column into the insertion point as the argument of the function. Figure 20 shows where the expression appears in the window.

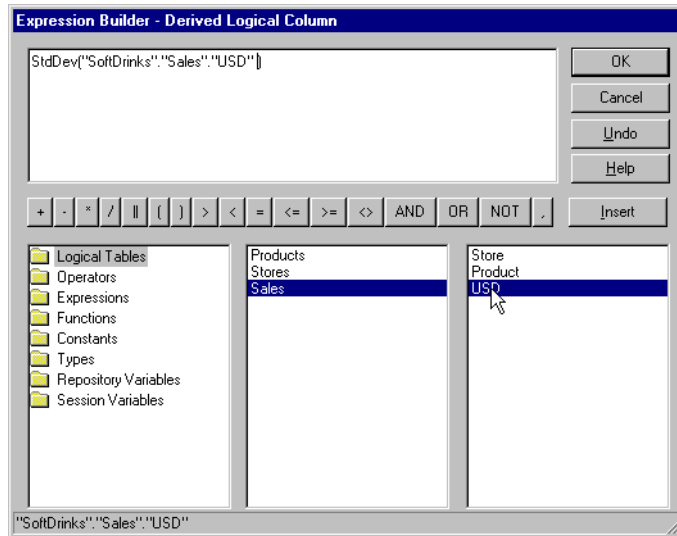


Figure 20. Example Logical Column Function in the Editing Pane

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 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.

Setting Up Aggregate Navigation **11**

Aggregate tables store precomputed results, which are measures that have been aggregated (typically summed) over a set of dimensional attributes. Using aggregate tables is a very popular 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. The aggregate navigation capability of the Siebel Analytics Server, however, allows queries to use the information stored in aggregate tables automatically, without query authors or query tools having to specify aggregate tables in their queries. The Siebel Analytics Server allows you to concentrate on asking the right business question; the server decides which tables provide the fastest answers.

For the Siebel Analytics Server to have enough information to navigate to aggregate tables, you need to configure certain metadata in the repository. This section provides details about and examples of how to set up the necessary metadata, and is divided into the following subsections.

- [“Specify the Aggregate Levels for Each Source” on page 250](#)
- [“Create Dimension Sources for Each Level of Aggregated Fact Data” on page 250](#)
- [“Specify Fragmentation Content” on page 253](#)
- [“Aggregate Table Fragments” on page 259](#)

Specify the Aggregate Levels for Each Source

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.

For the Siebel Analytics Server to be able to use the aggregate table to answer this query, it needs to know that the data in the aggregate table is stored at the product, store, and month levels. You configure this metadata in the Logical Table Source window.

Specify Content of Source

To use a source correctly, the Siebel Analytics Server needs to know what each source contains. You can specify the content of a source in the Content tab of the Logical Table Source dialog box. For more information see, [“Defining Content of Sources” on page 141](#).

WHERE Clause Filter

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.

Create Dimension 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.

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

- A source for the Product logical table with the following content specification:

By logical level: ProductDimension.ProductLevel

or

By column: Product.Product_Name

- A source for the Store logical table with the following content specification:

By logical level: StoreDimension.StoreLevel

or

By column: Store.Store_Name

- A source for the Time logical table with the following content specification:

By logical level: TimeDimension.MonthLevel

or

By column: Time.Month

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.

Although you have a choice to specify aggregate content by logical level or column, it is recommended that you use logical levels exclusively.

This source content information tells the Siebel 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 an aggregate fact table source and the corresponding aggregate dimension table source and show the physical diagram (selected tables only). Verify there is a join from the tables in the logical dimension table source to the tables in the logical fact table source.

Creating Sources for Each Logical Table

Use this procedure to create sources for logical tables.

To create the sources for each logical table

- 1** In the Business Model and Mapping layer of the Administration Tool, select the Sources folder under the logical table and select New Logical Table Source from the right-click menu.
- 2** In the General tab of the Logical Table Source window, type a name for the source.
- 3** Click Add to select the physical tables that populate the source.

For example, if the Product_Name column comes from a physical table in a database named *OrderEntry*, navigate to that table in the Browse window, select the table and click Select. This adds the table to the Logical Table Source window.

- 4** Map any columns in the logical to physical mapping area of the window.
Any physical columns with the same names as the logical columns are mapped automatically.
- 5** In the Content tab of the Logical Table Source window, describe the content specification of the source.
- 6** Set any necessary WHERE clause constraints.
- 7** Repeat this process.

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:

```
logicalColumn IN <valueList1>
```

Fragment *n*:

```
logicalColumn IN <valueListN>
```

Single Column, Range-Based Predicates

Fragment 1:

```
logicalColumn >= valueof(START_VALUE) AND logicalColumn <  
valueof(MID_VALUE1)
```

Fragment 2:

```
logicalColumn >= valueof(MID_VALUE1) AND logicalColumn <  
valueof(MID_VALUE2)
```

Fragment *n*:

```
logicalColumn >= valueof(MID_VALUEN-1) AND logicalColumn <  
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 15, “Using Variables in the Analytics Server Repository.”](#)) If you use repository values in your expression, note that the following construct will not work for Fragment 2:

```
logicalColumn >= valueof(MID_VALUE1)+1 AND logicalColumn <
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:

```
logicalColumn >= valueof(START_VALUE) AND logicalColumn <
valueof(MID_VALUE1)
```

Fragment 2:

```
logicalColumn >= valueof(MID_VALUE2) AND logicalColumn <
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:

```
<logicalColumn1 predicate> AND <logicalColumn2 predicate > ...
AND <logicalColumnM predicate>
```

Fragment *n*:

```
<logicalColumn1 predicate> AND <logicalColumn2 predicate > ...
AND <logicalColumnM predicate>
```

Ideally, all fragments will have predicates on the same M columns. If there is no predicate constraint on a logical column, the Siebel 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 255](#).

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):

```
EnterpriseModel.Period."Day" < VALUEOF("Snapshot Date") OR
EnterpriseModel.Period.MonthCode < VALUEOF("Snapshot Year Month")
OR
EnterpriseModel.Period."Year" < VALUEOF("Shapshot Year") OR
EnterpriseModel.Period."Year" = VALUEOF("Shapshot Year") AND
EnterpriseModel.Period."Month in Year" < VALUEOF("Snapshot
Month") OR
EnterpriseModel.Period."Year" = VALUEOF("Shapshot Year") AND
EnterpriseModel.Period."Monthname" IN ('Mar', 'Feb', 'Jan')
```

Fragment 2 (Current):

```
EnterpriseModel.Period."Day" >= VALUEOF("Snapshot Date") OR
EnterpriseModel.Period.MonthCode >= VALUEOF("Snapshot Year
Month") OR
EnterpriseModel.Period."Year" > VALUEOF("Shapshot Year") OR
```

```
EnterpriseModel.Period."Year" = VALUEOF("Shapshot Year") AND  
EnterpriseModel.Period."Month in Year" >= VALUEOF("Snapshot  
Month") OR
```

```
EnterpriseModel.Period."Year" = VALUEOF("Shapshot Year") AND  
EnterpriseModel.Period."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("Shapshot Year")  
OR
```

```
Track 4 EnterpriseModel.Period."Year" = VALUEOF("Shapshot Year")  
AND EnterpriseModel.Period."Month in Year" < VALUEOF("Snapshot  
Month") OR
```

```
Track 5 EnterpriseModel.Period."Year" = VALUEOF("Shapshot 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 Siebel 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 Siebel 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 Siebel 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("Shapshot Year") OR  
  
EnterpriseModel.Period."Year" = VALUEOF("Shapshot Year") AND  
EnterpriseModel.Period."Month in Year" <= VALUEOF("Snapshot  
Month") OR
```

```
EnterpriseModel.Period."Year" = VALUEOF("Shapshot Year") AND  
EnterpriseModel.Period."Monthname" IN ('Apr', 'Mar', 'Feb',  
'Jan')
```

Fragment 2 (Current):

```
EnterpriseModel.Period."Day" >= VALUEOF("Snapshot Date") OR  
  
EnterpriseModel.Period.MonthCode >= VALUEOF("Snapshot Year  
Month") OR  
  
EnterpriseModel.Period."Year" > VALUEOF("Shapshot Year") OR  
  
EnterpriseModel.Period."Year" = VALUEOF("Shapshot Year") AND  
EnterpriseModel.Period."Month in Year" >= VALUEOF("Snapshot  
Month") OR  
  
EnterpriseModel.Period."Year" = VALUEOF("Shapshot Year") AND  
EnterpriseModel.Period."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):

```
Marketing."Order Status"."Order Status" IN 'Shipped', 'Canceled')  
AND
```

```
(Marketing.Calendar."Calendar Date" <= VALUEOF("Snapshot Date")  
OR  
Marketing.Calendar."Year" <= VALUEOF("Snapshot Year") OR  
Marketing.Calendar."Year Month" <= VALUEOF("Snapshot Year Month")
```

Fragment 2 (Current):

```
Marketing."Order Status"."Order Status" IN ('Shipped',  
'Canceled') AND  
  
(Marketing.Calendar."Calendar Date" > VALUEOF("Snapshot Date") OR  
Marketing.Calendar."Year" >= VALUEOF("Snapshot Year") OR  
Marketing.Calendar."Year Month" >= VALUEOF("Snapshot Year Month")  
  
OR Marketing."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 Siebel 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 21, 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.

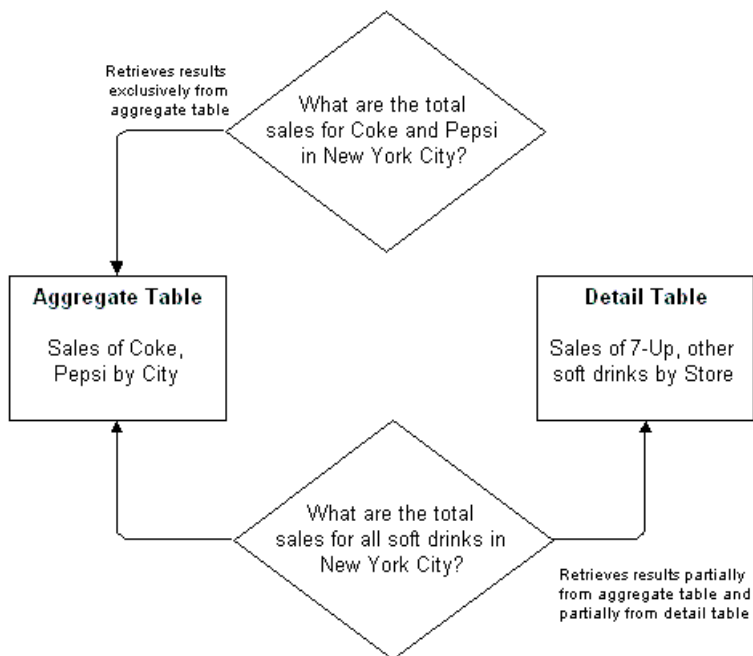


Figure 21. Aggregating Information

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 Siebel 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) looks like 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) looks like the following:

```
SoftDrinks.Products.Product IN ('Coke', 'Pepsi')
```

This content specification tells the Siebel 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 253](#).

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 which 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.

Setting Up Aggregate Navigation

Aggregate Table Fragments

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 22 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.

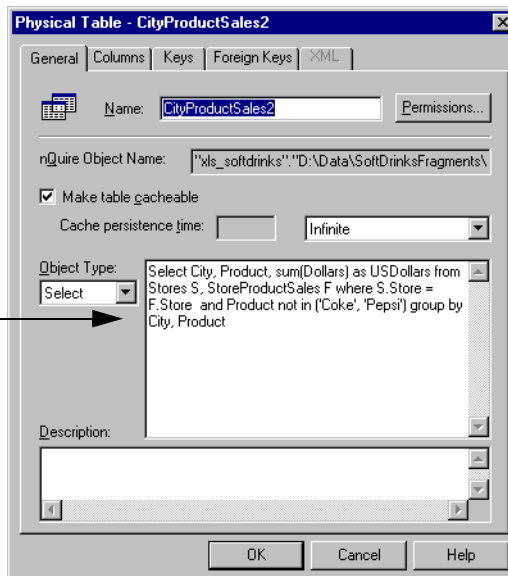


Figure 22. Example Physical Table Definitions

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:

```
GeographyDim.CityLevel, ProductDim.ProductLevel
```

This tells the Siebel Analytics Server this source has data at the city and product level.

The Fragmentation content specification might be:

```
SoftDrinks.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 Advanced 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 23](#).

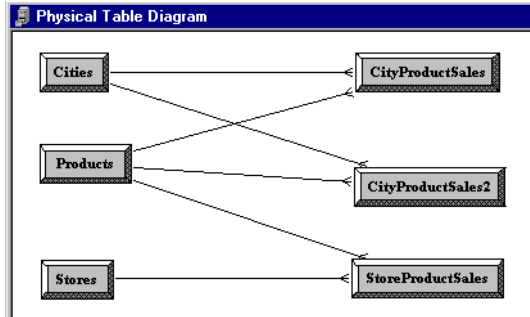


Figure 23. Example Physical Joins

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.

Administering the Query Environment **12**

The Siebel 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.

Starting the Siebel Analytics Server

The Siebel Analytics Server needs to be running before any queries can be processed. You can start the server in any of these ways:

- In Windows NT or Windows 2000, from the Control Panel Services applet.
- In Windows, by configuring the server for automatic startup.
- In UNIX, by running the Siebel Analytics Server startup script.

Starting the Server from the Services Applet in Windows

This procedure requires your Windows NT or Windows 2000 user ID to be a member of the Windows Administrators group on the local machine in which the Siebel Analytics Server is installed.

To start Siebel Analytics Server

- 1** On the machine in which the 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 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 NQSConfig.INI file during the startup operation.

When startup is complete, the startup message goes away and the status of the service in the Services applet 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, and any errors that occurred.

In the event that startup does not complete, check to make sure that there are no errors in the NQSConfig.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 Siebel Analytics Server to start automatically when Windows NT or Windows 2000 starts.

To configure the Siebel Analytics Server for automatic startup

- 1 On the machine in which the 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 Startup.

The Service dialog box for the Siebel Analytics Server appears.

- 4 Select Automatic for the Startup Type and click OK.

The Siebel Analytics Server service will now start automatically when Windows starts.

Running the Siebel Startup Script in UNIX

Start the Siebel 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 Siebel Analytics Server Runs

In Windows, 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 Siebel Analytics Server runs, you can do so from the Control Panel Services applet.

To change the user ID

- 1 On the machine in which the Siebel 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 Startup.

The Service dialog for the Siebel Analytics Server appears.

- 4 In the Log On As portion of the Services dialog box, select the option This Account, and then click the button to the right of the text box.

The Add User dialog box appears.

- 5 Select the user account in which you want the Siebel 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, which 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 Siebel Analytics Server software installation folder. You can use a text editor to view this file.
- In UNIX, run `/usr/sbin/syslogd` and look for any system and Siebel 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 NQServerConfig.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 Server

You can stop the Siebel Analytics Server in any of the following ways.

- In Windows, from the Windows NT or Windows 2000 Control Panel Services applet.
- In Windows, from an MS-DOS prompt.
- In UNIX, by running the Siebel Analytics Server shutdown script.
- By logging into a repository with the Administration Tool in online mode using a nonclustered DSN.

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 canceled.

Shutting Down the Server from the Services Applet in Windows

The following procedure explains how to shut down the Siebel 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 Siebel 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. When you are asked to confirm this, click Yes. 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 Siebel Analytics Server software installation folder.

Shutting Down the Server from a Command Prompt in Windows

Use this procedure in Windows to shut down the Siebel Analytics Server from a Command prompt.

To shut down the Siebel Analytics Server from a Windows command prompt

- Type the following command in the machine in which the Siebel Analytics Server is running:

```
nqsshutdown {-d <data_source_name> -u <user ID> -p <password>}
```

where:

<i>data_source_name</i>	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 a Siebel Analytics user who is a member of the Siebel Analytics ServerAdministrators group. Only users defined as Siebel 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 Siebel Analytics Server even if the server is participating in a cluster.

<i>user ID</i>	The user to connect to the Siebel Analytics Server to perform the shut down operation.
----------------	--

<i>password</i>	The password for the user ID connecting to the Siebel Analytics Server to perform the shut down operation.
-----------------	--

When shutdown is complete, a message indicating this is displayed in the Command window.

Running the Siebel Analytics Server Shutdown Script in UNIX

Stop the Siebel 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 Server Using the Administration Tool

The following procedure explains how to shut down the server using the Administration Tool.

NOTE: To shut down the server, the DSN has to log in as a user that has Siebel Analytics Server 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 into 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 Siebel Analytics Server instances offline. For more information, see [“Using the Cluster Manager” on page 353](#).

Getting Users to Connect to the Server

Users need to set up a data source to connect to a business model within a Siebel Analytics Server repository. For information on setting up DSN connections, see [“Configuring Siebel Analytics ODBC Data Source Names \(DSNs\)” on page 317](#).

You can also run the Administration Tool from a remote machine, either in online mode or offline mode.

Administering the Query Log

The Siebel Analytics Server provides a facility for logging query activity at the individual user level. Logging is intended for quality assurance testing, debugging, and for use by Siebel Technical Support. In production mode, query logging is normally disabled.

The query log file is named the NQQuery.log file. This file is in the Log subdirectory in the Siebel Analytics installation folder.

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 Siebel Systems 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 `NQConfig.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 Siebel Analytics Server for the change to take effect. For the syntax of the `USER_LOG_FILE_SIZE` parameter, see *Siebel Analytics 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. 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 Siebel Analytics Server administrators.

NOTE: Logging levels greater than 2 should be used only with the assistance of Siebel Technical Support.

The logging levels are described in [Table 24](#).

Table 24. 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>

Table 24. Logging Levels

Logging Level	Information That Is Logged
Level 3	Logs everything logged in Level 2. Additionally, logs the logical query plan. Do not select this level without the assistance of Siebel Technical Support.
Level 4	Logs everything logged in Level 3. Additionally, logs the query execution plan. Do not select this level without the assistance of Siebel Technical Support.
Level 5	Logs everything logged in Level 4. Additionally, logs intermediate row counts at various points in the execution plan. Do not select this level without the assistance of Siebel Technical Support.
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 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 Siebel Analytics Server 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. If not specified, the active log file is read.
<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 Siebel 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 Siebel 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 284](#).

NOTE: Siebel Analytics 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 Siebel 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, which 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 Siebel 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 which tables 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 Siebel 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 Siebel Analytics Server tracks usage at the detailed query level.

When you enable usage tracking, statistics for every query are written to a usage tracking log file or inserted into a database table. Siebel recommends direct insertion into a database table. For more information, see *Siebel Analytics Installation and Configuration Guide*.

Administering Direct Insertion for Usage Tracking

To set up direct insertion for usage tracking, use the guidelines in this section.

Enabling Direct Insertion

In the Usage Tracking section of the NQSConfig.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 Siebel 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 Siebel 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 Siebel 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 Siebel 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.

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 Siebel Analytics software installation folder.

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 Siebel 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 Siebel 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 Siebel Analytics Server 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 linefeed delimits the end of each row of data.

The schema is described in [Table 25](#). For more information about the contents of each column, see [“Description of the Usage Tracking Data” on page 478](#).

Table 25. 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 db queries	Integer (see following Note)	4	Yes

Table 25. Usage Tracking Output File Format

Column Number	Column Name	Data Type	Max Data Size	Nullable
17	Cumulative db time (secs)	Integer (see following Note)	4	Yes
18	Cumulative db rows	Integer (see following Note)	4	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. In this regard, 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 477](#).

Performance Considerations

When usage tracking is enabled, the Siebel 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 NQSConfig.INI file (see *Siebel Analytics 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 Siebel Analytics Server.

The Siebel 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.

Collecting More Detailed Information About Queries

To collect more detailed information about queries, you can set the parameters in the Server Query Statistics section of the NQSConfig.INI file. When usage tracking is enabled, the server query statistics parameters define default values for the collection of more detailed statistics for logical queries, as well as for physical queries issued by the Siebel Analytics Server to back-end databases. Some of this information is collected when usage tracking is enabled; the server query statistics provide more detailed information.

See *Siebel Analytics Installation and Configuration Guide* for details about the parameters. You can also control the collection of these statistics through the Query Statistics utility in the Administration Tool.

The name of the file that holds the server query statistics is the NQQueryStats.log, located in the QueryStats folder in the Siebel Analytics Server software installation folder. Entries in the NQQueryStats.log file can be viewed using a text editor.

Server Session Management

The Session Manager is used in online mode to monitor activity. The Session Manager shows all users logged into the, all current query requests for each user, and variables and their values for a selected session. Additionally, the Siebel Analytics Server 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 Siebel 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 Siebel Analytics Servers 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 26 and Table 27 describe the columns in the Session Manager windows.

Table 26. Fields in the Session Window

Column Name	Description
Session ID	The unique internal identifier that the Siebel Analytics Server assigns each session when the session is initiated.
Query Server	The Siebel Analytics Server that serviced the request.
User	The name of the user connected.
Client Type	The type of client connected to the server.
Catalog	The name of the presentation catalog to which the session is connected.
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 Siebel Analytics Server.
Last Active Time	The timestamp of the last activity on the session.

Table 27. Fields in the Request Window

Column Name	Description
Session ID	The unique internal identifier that the Siebel Analytics Server assigns each session when the session is initiated.
Query Server	The Siebel Analytics Server that serviced the request.
Request ID	The unique internal identifier that the Siebel 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.
Status	The status of the query. See <i>Siebel Analytics Server Administration Tool Online Help</i> for the meanings of the different statuses.
Description	Not used.

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.
- 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 an 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 an 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 Siebel Analytics Server.

NQSConfig.INI File Parameters

The NQSConfig.INI file contains configuration and tuning parameters for the Siebel 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 information on the NQSConfig.INI file parameters, see *Siebel Analytics 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 Siebel 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 Aggregate Navigation” on page 249](#).

Query Caching

Enabling query caching causes the Siebel Analytics Server to store query results for reuse by subsequent queries. Caching can dramatically improve the apparent performance of the system for users. For information on query caching concepts and setup, see [Chapter 13, “Query Caching in Siebel Analytics Server.”](#)

Tune and Index Underlying Databases

The Siebel 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 information on configuring query logging on your system, see [“Administering the Query Log” on page 272.](#)

Query Caching in Siebel Analytics Server **13**

Decision support queries sometimes require large amounts of database processing. The Siebel 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 Siebel Analytics Server.

About the Analytics Server Query Cache

The Siebel Analytics Server stores metadata about each stored query result. It uses the cache metadata to evaluate whether new queries can use results already stored in cache. If a query does qualify to use results stored in the cache, it is called a cache hit. The Analytics Server can use cache to answer queries at the same level or at a higher level of aggregation.

NOTE: The parameters to control query caching are located in the NQSConfig.INI file described in *Siebel Analytics Installation and Configuration 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 Siebel 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 Siebel 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 Siebel 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 Siebel Analytics Server processing overhead.

Security Enforced

Query caching enforces all security attributes that are set in the system. If a query is cached from a user whose configuration specifies database-specific logon IDs, the cache entry is only valid for that user. Similarly, if a query is cached, only users who have privileges to ask the query can read the query from the cache. For more information about security, see [Chapter 17, “Security in Siebel 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 trade offs 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 Siebel Analytics Server for query caching.

The parameters to control query caching are located in the NQSConfig.INI file, described in *Siebel Analytics Installation and Configuration Guide*.

Configuring the Cache Storage

The following items need configuration in the NQSConfig.INI file for cache storage:

- 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 `NQSConfig.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 information about cache hits, see [“Cache Hits” on page 298](#).

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.

See *Siebel Analytics Installation and Configuration Guide* for more information about this configuration parameter.

Metadata Filename

The cache metadata file stores information about queries stored in the cache. The Siebel Analytics Server creates and continually updates the metadata file in the location specified in the `METADATA_FILE` parameter in the `CACHE` section of the `NQSConfig.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 Siebel 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` NQSCONFIG.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 Analytics 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 295](#), 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 Siebel 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 specify a *cachable* attribute for each physical table; that is, if queries involving the specified table can be added to the cache to answer future queries. To enable caching for a particular physical table, select the table in the Physical layer of the Administration Tool and select the option Make table cachable in the General tab of the Physical Table properties dialog box. You can also use the Cache Persistence Time settings to specify how long the entries for this table should persist in the query cache. This is useful for OLTP data sources and other data sources that are updated frequently, potentially down to every few seconds.

After caching is enabled 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, which is updated every minute. You could use the Cache Persistence Time settings to purge the entries for that table every 59 seconds.

To disable caching for a particular physical table, select the table in the Physical layer of the Administration Tool and clear the option Make table cachable in the General tab of the Physical Table dialog box. After caching is disabled for a table, any query involving the table is not added to the cache.

Configure Siebel Analytics Server Event Polling Tables

Siebel Analytics Server event polling tables store information about updates in the underlying databases. An application (such as an application 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. For more information on event polling tables, see [“Setting Up Event Polling Tables on the Physical Databases” on page 304.](#)

Purging Cache Programmatically

The Analytics Server provides ODBC-extension functions for purging cache entries programmatically. Scripts that call these ODBC-extension functions must run under an administrator logon ID. Only administrators have the right to purge cache. 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 Siebel Analytics repository, execute the following call in a script:

```
Call SAPurgeCacheByTable( 'DBName', 'CatName', 'SchName',  
'TabName' );
```

- Nulls passed as input parameters to SAPurgeCacheByTable serve as wild cards. For example, specifying a database name but leaving the catalog, schema and table names null will direct the function to purge all entries associated with the specified database.

- Similarly, specifying a table name but leaving database, catalog and schema names null directs the function to purge all cache entries associated with the specified table name. If multiple databases, catalogs or schemas have tables with the specified name, the function will purge all cache entries associated with any of those tables.

CAUTION: Leaving all input parameters null directs the function to purge all cache entries associated with the repository.

- **SAPurgeAllCache.** Purges all cache entries. The following is an example of this call:

```
Call SAPurgeAllCache();
```

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 a knowledge of when cache hits occur.

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. 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 employeename, region
from employee, geography
where region in ('EAST', 'WEST')
```

qualifies as a hit on the following cached query:

```
select employeename, region
from employee, geography
where region in ('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 Siebel 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, unitsales 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 Siebel Analytics 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 server purges all cache for those business models.

Cache Event Processing with an Event Polling Table

The use of a Siebel 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 Siebel 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 Siebel 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 28 on page 304](#). 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 Siebel 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 Siebel Analytics Server Event Polling Table” on page 309](#)). 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 Siebel Analytics Server 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 Siebel 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 28](#); 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 28](#). Sample CREATE TABLE statements to create an event polling table are shown in “[Sample Event Polling Table CREATE TABLE Statements](#)” on [page 306](#).

Table 28. Event Polling Table Column Names

Event Table Column Name	Datatype	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.

Table 28. Event Polling Table Column Names

Event Table Column Name	Datatype	Description
DatabaseName	CHAR or VARCHAR	<p>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.</p> <p>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.</p>
CatalogName	CHAR or VARCHAR	<p>The name of the catalog where the physical table that was updated resides.</p> <p>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.</p>
SchemaName	CHAR or VARCHAR	<p>The name of the schema where the physical table that was updated resides.</p> <p>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.</p>
TableName	CHAR or VARCHAR	<p>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.</p> <p>Values cannot be NULL.</p>
Other	CHAR or VARCHAR	<p>Reserved for future enhancements. This column must be set to a NULL value.</p>

The Siebel 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.

To allow Siebel Analytics Server write access to the event polling table but not to any other tables in a database, you can create a separate physical database in the Physical layer of the Administration Tool with a privileged connection pool (privileged to delete from the table). Populate this privileged database with the event table. Setting up this type of polling database makes sure that the Siebel Analytics Server has 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 Siebel 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 Siebel Analytics Server.

To make the polling table active

- 1 Import the table to the Physical layer.
- 2 Include it in the group of Siebel 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. Click OK.

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 noncachable.

When a table has been registered as a Siebel Analytics Server event table, the table properties change. Registration as an event table removes the option to make the table cachable, as there is no reason to cache results from an event polling table.

Populating the Siebel Analytics Server Event Polling Table

The Siebel 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 Siebel 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 Siebel 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 Siebel Analytics Server. The default location for this file is the Log folder in the Siebel Analytics Server software installation folder. Log entries are self-explanatory and can be viewed using a text editor.
- When the Siebel 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 Siebel Analytics Server 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 Siebel Analytics Server software installation folder. For more information about user-level logging, see [“Administering the Query Log” on page 272](#).

Making Changes to a Repository

When you modify Siebel Analytics Server 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 a Siebel Analytics Server 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 a Siebel Analytics Server 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 313](#).

Using the Cache Manager

The Cache Manager provides Siebel Analytics Server 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 hand 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 29](#).

Table 29. Cache Options

Option	Description
Query Server	The Siebel 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 Siebel 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 milliseconds, needed to create the result set for this cache entry.
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.

Table 29. Cache Options

Option	Description
Full size	The total number of bytes stored in this cache entry's result set. This is the product of row count times row size and does not include any overhead.
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 Siebel 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 30](#) describes the information that appears in the Global Cache Information window.

Table 30. 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 NQSConfig.INI file.
Amount of space still available for cache storage use	The amount of space, in megabytes, still available for cache storage.

Table 30. Global Cache Information

Column	Description
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 NQSConfig.INI file.
Number of queries satisfied from cache since startup of Siebel Analytics Server	Cache hits, since the last time the Siebel Analytics Server was started.
Number of queries not satisfied from cache since startup of Siebel Analytics Server	Cache misses, since the last time the Siebel 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 Siebel 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 information about setting up an XML data source and specifying a refresh level in the Administration Tool, see [“Using XML as a Data Source for Analytics” on page 389](#).

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 Siebel 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.

Query Caching in Siebel Analytics Server

About the Refresh Interval for XML Data Sources

The Siebel Analytics Server provides the functionality for you to connect to it through many client tools and applications. This section describes how to configure ODBC data source names and provides information about connectivity using third-party tools.

Configuring Siebel Analytics ODBC Data Source Names (DSNs)

In a non-English environment, you cannot use a direct ODBC connection to Siebel Analytics Server. Only the Siebel Analytics Web client can connect directly to Siebel 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, the Data Sources (ODBC) icon is available as an Administrative Option.

- 2** Click Add in the ODBC Data Source Administrator.
- 3** Select the driver Siebel Analytics Server from the Create New Data Source dialog, and then click Finish.

The first Siebel Analytics Server 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 that the Siebel Analytics Server is running on.

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 Siebel 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 Siebel 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 `NQSConfig.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 Siebel Analytics Server has changed from Distributed component object model (DCOM) to TCP/IP. Siebel Systems will discontinue support for DCOM in a future release. For sites already running the Siebel 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 `NQSConfig.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 `NQSConfig.INI` file, or to the first repository listed if none of the entries is marked as the default.

- 12** Select the option Connect to Siebel 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 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 that the Siebel Analytics Server connects to. 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 a Connection Pool for All Data Sources” on page 93](#). 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 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 Server in the previous screen. The new password is stored in encrypted form in the repository.

Third-Party Tools and Relational Datasource Adapters

The Siebel 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 a Siebel Analytics Server repository. The metadata is used to establish physical table information in the Physical layer of the Administration Tool.

Metadata imports to a Siebel Analytics Server 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 XMLA data sources.

For the metadata import procedure, see [“Process of Creating the Physical Layer by Importing a Physical Schema”](#) on page 82.

Using Query and Reporting Tools

You can connect to the Siebel 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 Siebel Analytics ODBC driver and then using the Siebel 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, which can take advantage of the Siebel 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 Siebel Analytics Server as a data source will make these tools more valuable and will simplify the work entailed when using them.

ODBC Conformance Level

The Siebel 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 Analytics ODBC supports full scrollable cursors with static, dynamic, forward only, and key set driven cursors.

Siebel Analytics ODBC supports asynchronous and synchronous processing and cancellation.

Using Variables in the Analytics Server Repository **15**

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 describes the different classes of variables and gives examples of how they can be used.

Using the Analytics 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 Siebel 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”](#)
- [“Dynamic Repository Variables” on page 327](#)

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 a Siebel Analytics Server 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 Siebel 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 Siebel 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, which 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 Siebel 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 Siebel 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 Siebel 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 17, “Security in Siebel Analytics.”](#)

This section includes the following topics:

- [“Using System Session Variables”](#)
- [“Using Nonsystem Session Variables” on page 331](#)

For information about creating a new session variable, see [“Creating NewVariables” on page 331](#).

Using System Session Variables

System variables are session variables that the Siebel Analytics Server and Siebel Analytics Web use for specific purposes. System variables have reserved names, which cannot be used for other kinds of variables (such as static or dynamic repository variables, or for nonsystem session variables).

Table 31 describes the available system session variables. When using these variables in the 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 31. System Session Variables

Variable	Description
USER	Holds the value the user enters as his or her logon name.
GROUP	<p>Contains the groups that the user belongs to. These are used by both the Siebel Analytics Server and Siebel Analytics Web.</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 datatype column in a database table would be:</p> <pre>Sales US;Sales UK;QA and Dev;Doc</pre> <p>Note: The Siebel Analytics Web administrator needs to make sure that the names of Web groups are different from any user IDs who will log on to Siebel 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 Analytics Web.</p>
DISPLAYNAME	Used for Siebel 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 Siebel Analytics Server repository users (nondatabase users), this variable is populated with the user's full name.
PORTALPATH	Used for Siebel Analytics Web. It identifies the default dashboard the user sees when logging in (the user can override this preference after logged on).
LOGLEVEL	The value of LOGLEVEL (a number between 0 and 5) determines the logging level that the Siebel Analytics Server will use for the user's queries.
WEBGROUPS	Specifies additional groups specific to Siebel Analytics Web, if any. The use of Web groups provides a mechanism for more granular Web content control.

Table 31. System Session Variables

Variable	Description
REQUESTKEY	Used for Siebel Analytics Web. Any users with the same nonblank requestkey will share the same Web cache entries. This tells Siebel Analytics Web that these users have identical content filters and security in the Siebel 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 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 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 Siebel Answers. If the delivery feature of Siebel 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 Analytics Web.

Using Nonsystem Session Variables

The procedure for defining nonsystem session variables is the same as for system session variables. When using these variables in the 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.

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.

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, chose Action > New, and then chose 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 From the Type drop-down list, select the type of variable.
 - For repository variables, choose Static Repository Variable or Dynamic Repository Variable.
 - For session variables, choose Session Variable.
- 5 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. See [“SQL Logical Operators” on page 427](#) for information on creating the value.
 - 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 (').

- 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 information about constructing a new initialization block, see [“Creating and Editing Initialization Blocks” on page 336](#).

- 7 Click OK.

About Initialization Blocks

Initialization blocks are used to initialize dynamic repository variables, system session variables, and nonsystem session variables. (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: When you open the Initialization Block dialog box for editing in online mode, you will check out the initialization block object. While the initialization block is checked out, the Siebel Analytics Server may continue to refresh the value of dynamic variables refreshed by this initialization block, depending on the refresh intervals that are set. If you later check the initialization block back in, you will reset the value of the dynamic variables to the values shown in the Default initializer. If you do not want this to occur, use the Undo Check Out option.

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 Siebel Analytics Server will follow to execute the query and periodically refresh the value of the variable. If you stop and restart the Siebel Analytics Server, the server will automatically execute the SQL in repository variable initialization blocks, reinitializing the repository variables.

The Siebel 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 Siebel Analytics Server 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 Siebel Analytics Server software installation folder. For more information about user-level logging, see [“Administering the Query Log” on page 272](#).

Initializing Session Variables

As with dynamic repository variables, session variables obtain their values from initialization blocks. Unlike dynamic repository variables, however, session variables do not get updated at scheduled time intervals. Instead, the Siebel 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 Siebel Analytics Server logs all SQL queries issued to retrieve session variable information in the NQQuery.log file when the session's user ID is set to 2 or higher. The default location for the NQQuery.log file is the Log folder in the Siebel Analytics Server software installation folder. For more information about user-level logging, see [“Administering the Query Log” on page 272](#).

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.

Example: Suppose 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 336](#)).

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 Siebel 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 Siebel 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 Siebel 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.

Use the instructions in this section to create a new initialization block or to edit properties of an existing initialization block.

To create or edit an initialization block

- 1 From the Administration Tool menu bar, select Manage > Variables.
- 2 Choose Action > New > Initialization Block.
- 3 In the General tab, type a name for the block. (The NQ_SYSTEM initialization block name is reserved.)
- 4 Select one of the following options:
 - associate with repository variable
 - associate with session variable

- 5** If you selected *associate* with session variable in the preceding step, 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 334](#).

- a** Select the Row-Wise Initialization check box.
- b** Select the Cache Variables check box to direct the Siebel Analytics Server to store the results of the query in a main memory cache.

If you select this option, the Siebel 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 should not choose this option.

- c** Type the SQL initialization string needed to populate the variables.

- 6** Specify a refresh interval in the Refresh time area.

- a** From the Start on drop-down list, select a date and time.

You can specify the day, the month, the day of the month, the year, and the hours, minutes and seconds.

- 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.

NOTE: The Refresh time area only applies to repository variables. It is unavailable for session variables.

- 7** In the Data Source Connection area, select Database, or LDAP if you are authenticating LDAP users.

TIP: You should test the SQL using the Test button or an SQL tool such as the Siebel Analytics Client utility. Use the same DSN or one set up identically to the DSN in the specified connection pool.

- 8** If you selected 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.
 - a** To select the connection pool associated with the database where the target information is located, click Browse.
 - b** In the Browse dialog box, select the connection pool and click OK.

NOTE: If you do not select a connection pool before typing the initialization string, you will receive a message prompting you to select the connection pool. You can click the Test button to test the connection.

- 9** If you selected Database in the Data Source Connection area, in the Initialization Block dialog box, type the initialization string to use.

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.

- For sites *not* using Siebel Delivers, a sample SQL statement follows:

```
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 Siebel Analytics Server, substituting valid values for the USER and PASSWORD variables.

- For sites that *are* using Siebel Delivers, a sample SQL statement follows:

```
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 Siebel 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 Siebel Analytics Server substituting valid values for the USER and PASSWORD variables and removing the NQS_PASSWORD_CLAUSE clause.

- For initialization strings that are database-specific, select the database type from the drop-down list above the Initialization String text box.

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.

- 10** If you selected LDAP in the Data Source Connection area, click Browse to select the LDAP Server this block will use, or click New... to open the General tab of the LDAP Server dialog box and create an LDAP Server. Click OK to return to the Initialization Block dialog box.

- 11** (Optional) Click Test in the Data Source Connection area.

The Results dialog box lists the variables and their values.

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 Siebel 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.

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 Use the drag-and-drop feature to reposition the column, or click the Up and Down buttons.

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 either double-click it or click the Edit button 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 Siebel Analytics Server.

Initialization When the Siebel Analytics Server Starts

If you stop and restart the Siebel 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.

Using Variables in the Analytics Server Repository

About Initialization Blocks

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 Siebel Analytics Servers in a network domain to act as a single server. Servers in the cluster share requests from multiple Siebel Analytics clients, including Siebel Analytics Answers and Siebel Analytics 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 `nqscmgateway.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 an 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 an 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. All servers must have read and write access to this directory.

In the NQConfig.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 Analytics 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 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 Analytics ODBC data source for clustering.
- 5** Copy the NQClusterConfig.INI file to the Cluster Controllers and Analytics Servers in the cluster.
- 6** Start the machines in the cluster.

For detailed information about installing and configuring this feature, see *Siebel Analytics 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 Analytics Installation and Configuration Guide*.

Setting Parameters in the NQSConfig.INI File

In the Server section of the NQSConfig.INI file, there are three parameters you need to set for a Siebel 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 Analytics software installation folder. For more information about these parameters, see *Siebel Analytics 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 Analytics Server 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 which machine acts as the primary Cluster Controller.
- **SECONDARY_CONTROLLER**—Specifies which machine 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 Analytics installation folder. For more information about these parameters, see *Siebel Analytics Installation and Configuration Guide*.

Configuring the Siebel Analytics ODBC Data Source Name

All clients, including 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 Server DSN Configuration wizard, described in [“Connectivity and Third-Party Tools” on page 317](#).

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 Analytics 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 Servers.

To manually start an 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 When CLUSTER_PARTICIPANT is set to YES in its NQSConfig.INI file, each Analytics Server then reads its NQClusterConfig.INI file. Cluster Controllers also read this file. If an 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 NQClusterConfig.INI file. If a machine is hosting both a Siebel 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 system verifies the presence of an active, operational Cluster Controller. If the machine hosting the primary Cluster Controller is not available, the system 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 system 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 `NQConfig.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 Analytics 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 32](#) describes categories that may appear.

Table 32. Status View Columns

Column	Description
Name	The name of the machine hosting the Siebel 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 Siebel 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 - An 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 32. 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. The Analytics Server is being quiesced. 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	<p>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.</p>
Last Reported Time	<p>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.</p>
Sessions	<p>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.</p>

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 33](#) describes categories that may appear.

Table 33. 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 an 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	Total number of bytes stored in this cache entry's result set. This is the product of row count times row size and does not include any overhead.
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 34](#).

- 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 34](#) describes the information that appears in the Session window.

Table 34. Session Window Columns (Top Window)

Column	Description
Session ID	Unique internal identifier that 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 Analytics Server 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 35](#) describes the information that appears in the Request window.

Table 35. 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.

Table 35 describes the information that appears in the Request window.

Table 35. Request Window Columns (Bottom Window)

Column	Description
Start Time	Time of the initial query request.
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 a Siebel Analytics Server administrator canceled 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.
Description	Currently unused.

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.

Security in Siebel Analytics 17

The Siebel Analytics Server provides secure access control at any level of granularity. This section provides information about the functionality available in the Siebel Analytics Server that allows Siebel Analytics Administrators to set permissions for business models, tables, and columns; specify special database access for each user; and use filters to limit the data accessible by users.

Analytics Security Manager

The 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 on 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: For information about configuring security for Siebel Analytics applications, see *Siebel Analytics Installation and Configuration Guide*. However, you should read this section first to gain an understanding about the basics of security and setting up authentication.

The Siebel 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 Siebel Analytics Server uses the Hyper Text Transport Protocol Secure (HTTPS) standard to send the information to a secure port on the Web server. From the Web server, the information is passed through ODBC to the Siebel Analytics Server, using Triple DES (Data Encryption Standard). This provides an extremely high level of security (168 bit), preventing unauthorized users from accessing data or 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 Analytics metadata repository.

This section includes the following topics:

- [“Working with Users”](#)
- [“Working with Groups” on page 365](#)
- [“Importing Users and Groups from LDAP” on page 371](#)

Working with Users

User accounts can be defined explicitly in a Siebel Analytics Server 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 Siebel Analytics Server for a session to take place, unless the Siebel Analytics Server administrator has configured the system to bypass Siebel Analytics Server security. (See [“Bypassing Siebel Analytics Server Security” on page 383](#) for details.)

Users defined explicitly in a repository can access business models in that repository, but they cannot span repositories.

The Siebel Analytics Server Administrator user account is created automatically when a repository is created and cannot be deleted. For more information on the Administrator user account, see [“Siebel Analytics Server Administrator Account” on page 364](#).

This section includes the following topics:

- [“Adding a New User to a Repository” on page 362](#)
- [“Siebel Analytics Server Administrator Account” on page 364](#)

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. (See [“Setting a Logging Level” on page 273](#) for more information on query logging.)
- 6** Click OK.

This creates a new user with no rights granted to it.
- 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 Siebel Analytics Server, the logon IDs in the DSN are used if the Siebel Analytics Server 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 a Connection Pool for All Data Sources” on page 93](#).

- 11 Set up any query permissions for the user. For information, see [“Managing Query Execution Privileges” on page 384](#).

Siebel Analytics Server Administrator Account

The Siebel Analytics Server Administrator account (user ID of Administrator) is a default user account in every Siebel Analytics Server 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 Siebel 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 Siebel 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 Siebel Analytics repository, that is, the repository shipped with Siebel 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 NQSSConfig.ini file using the MINIMUM_PASSWORD_LENGTH setting.

Working with Groups

The Siebel 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 Siebel 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, Siebel Analytics Server groups can allow access to objects. Additionally, Siebel 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 329](#).

This section includes the following topics:

- [“Predefined Administrators Group”](#)
- [“Defined Groups” on page 366](#)
- [“Group Inheritance” on page 366](#)
- [“Adding a New Group” on page 369](#)
- [“Viewing Member Hierarchies” on page 370](#)

Predefined Administrators Group

The Siebel Analytics Server has one predefined group, the Siebel Analytics Server Administrators group. Members of this group have the authority to access and modify any object in a repository. The predefined Siebel Analytics Server Administrator user ID is automatically a member of the Siebel 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 a Siebel Analytics Server repository. Each group can contain explicitly granted privileges or privileges granted implicitly using membership in another group. For information on setting up a group, see [“Adding a New Group” on page 369](#).

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, which 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, which explicitly denies access to TableA. The resultant privilege for User1 is to read TableA, as shown in [Figure 24](#).

Because privileges granted directly to the user take precedence over those granted through groups, User1 has the privilege to read TableA.

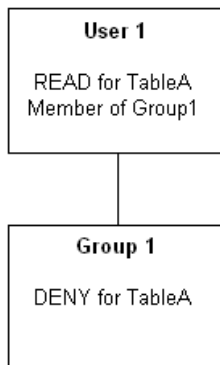


Figure 24. User Privileges and Group Privileges

Example 2

Consider the situation shown in [Figure 25](#).

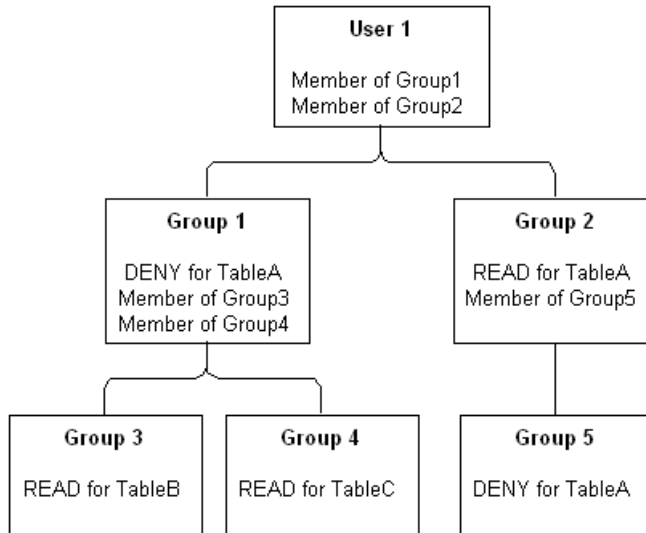


Figure 25. Privileges Example

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 384](#).

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 Siebel 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 [“LDAP Authentication” on page 376](#).

This section includes the following topics:

- [“Configuring an LDAP Server”](#)
- [“Importing Users from LDAP” on page 373](#)
- [“Synchronizing Users and Groups with LDAP” on page 374](#)

NOTE: If a user exists in both the repository and in LDAP, the local repository user definition takes precedence. This allows the Siebel Analytics Server Administrator to reliably override users that exist in an external security system.

Configuring an LDAP Server

The following procedure explains how to configure LDAP authentication for the repository.

NOTE: The Siebel 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** Display the security window by selecting Manage > Security.
- 3** Select Action > New > LDAP Server.

- 4 Type the information requested in the LDAP Server dialog box.
 - **Host name.** The name of your LDAP server.
 - **ADSI.** (Active Directory Service Interfaces) A type of LDAP server. If you select the ADSI check box, Bind DN and Bind password are required.
 - **Port number.** The default LDAP port is 389.
 - **LDAP version.** LDAP 2 or LDAP 3. 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 associated user password required to bind to the LDAP server.

If these two entries are left blank, then *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 requested information.

NOTE: The Siebel Analytics Server maintains an authentication cache in memory, which 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 is connecting to an LDAP server for import purposes or the Siebel Analytics Server is connecting to an LDAP server for user authentication, the connection will time out after this interval.
- **Cache refresh interval.** The interval at which the authentication cache entry for a logged on user will be refreshed.
- **Number of Cache Entries.** The maximum number of entries in the authentication cache, preallocated at Siebel Analytics Server startup time. If the number of users exceeds this limit, cache entries are replaced using the LRU algorithm.

If this value is 0, then cache is disabled.

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** Display the security window by selecting Manage > Security.
- 3** 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 Analytics users for testing.

Authentication Options

Authentication is the process by which a system verifies, through the use of a user ID and password, that a user has the necessary permissions and authorizations to log in and access data. The Siebel Analytics Server authenticates each connection request it receives.

The Siebel Analytics Server supports the following authentication schemes:

- [“Operating System Authentication”](#)
- [“LDAP Authentication” on page 376](#)
- [“External Table Authentication” on page 378](#)
- [“Database Authentication” on page 381](#)
- [“Siebel Analytics Server Internal Authentication” on page 382](#)

Operating System Authentication

The Siebel Analytics Server supports Windows NT and Windows 2000 Unified Logon. If a user is configured on a trusted Windows domain, a Siebel Analytics Server user of the same name does not need to be authenticated by the Siebel Analytics Server; the user has already been authenticated by Windows and is therefore trusted to log in to the Siebel Analytics Server.

When operating system authentication is enabled, users connecting to the Siebel Analytics Server should not type a user ID or password in the logon prompt. If a user enters a user ID and (optionally) a password in the logon prompt, that user ID and password overrides the operating system authentication and the Siebel Analytics Server performs the authentication.

NOTE: Operating system authentication cannot be used with Analytics Web. It can only be used with ODBC client applications.

To configure operating system authentication

- 1** Enable operating system authentication by specifying the `PERFORM_OS_AUTHENTICATION = YES` security option in the `NQConfig.INI` file. (See *Siebel Analytics Installation and Configuration Guide* for additional information.)
- 2** Create a user in your repository named identically to the user ID in the trusted Windows domain.
- 3** Assign the group membership and rights you want the user to have.

The user is now trusted to log in without any further authentication, provided the request comes from a Windows NT or Windows 2000 session logged in on the trusted domain as the trusted user ID. In other words, users with identical Windows and Siebel Analytics Server user IDs do not need to submit a password when logging in to the Siebel Analytics Server from a trusted domain.

For more information about Windows and Windows trusted domains, see your Windows NT or Windows 2000 security documentation.

LDAP Authentication

Instead of storing user IDs and passwords in a Siebel Analytics Server repository, you can have the Siebel Analytics Server pass the user ID and password entered 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 Siebel Analytics Server with other information, such as the user display name (used by Siebel 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 which get passed to Siebel Analytics *session variables* during the process of user authentication. For more information about session variables, see [“About Session Variables” on page 328](#).

Setting Up LDAP Authentication

LDAP authentication uses Siebel Analytics session variables, which you define using the Variable Manager of the Administration Tool. For more information about the Variable Manager, see [“Using the Analytics Variable Manager” on page 325](#).

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 329](#).

To configure LDAP authentication, you define a system variable called USER and associate it with an LDAP *initialization block*, which is associated with an LDAP server. Whenever a user logs into the Siebel 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 371](#).

NOTE: The presence of a defined session system variable USER determines that external authentication is done. 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 details, see [“External Table Authentication” on page 378](#).

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.
The Session Variable - User dialog box appears.
- 4** 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 273](#) for more information.

External Table Authentication

Instead of storing user IDs and passwords in a Siebel Analytics Server 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 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 [“Database Authentication” on page 381](#), and [“Order of Authentication” on page 383](#) for additional details.

Setting Up External Table Authentication

External table authentication uses Siebel Analytics session variables, which you define using the Variable Manager of the Administration Tool. For more information about the Variable Manager, see [“Using the Analytics Variable Manager” on page 325](#).

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*, which 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 on session variables, see [“About Session Variables” on page 328](#).

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 details, see “[LDAP Authentication](#)” on page 376.

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.

The Initialization Block dialog box appears.

- 5** 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 Siebel 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.

Database Authentication

The Siebel 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 Siebel Analytics Server. Unlike operating system authentication, this authentication can be applied to Siebel Analytics Web users.

NOTE: Siebel Delivers does not work with database authentication.

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 [“External Table Authentication” on page 378](#) and [“Order of Authentication” on page 383](#) for additional details.

Database authentication requires the user ID to be stored in the Siebel Analytics Server repository.

To use 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 NQSCONFIG.INI file. (See *Siebel Analytics 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 NQSCONFIG.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 Siebel 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 Siebel Analytics Server issues a message to the user indicating an invalid user ID or password.

Siebel Analytics Server Internal Authentication

You can maintain lists of users and their passwords in the Siebel Analytics Server repository using the Administration Tool. The Siebel 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 383](#) for additional information.

Siebel Analytics Server User IDs and Passwords

The Siebel Analytics Server user IDs are stored in nonencrypted form in a Siebel Analytics Server repository and are case-insensitive. Passwords are stored in encrypted form and are case-sensitive. The Siebel 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 Siebel Analytics Server repository.

To change a user password

- 1 Select Manage > Security.
The Security Manager appears.
- 2 Select Users in the Security tree.
- 3 In the right pane, right-click the user whose password you want to change.

- 4 Select Properties from the shortcut menu.
The User dialog box appears.
- 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 NQSConfig.INI file. (See *Siebel Analytics Installation and Configuration Guide* for details.) OS authentication is also not used for Siebel Analytics Web users.

The Siebel Analytics Server populates session variables using the initialization blocks in the desired order, which 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.

Siebel Analytics Server internal authentication (or, optionally, database authentication) occurs only after these other possibilities have been considered.

Bypassing Siebel Analytics Server Security

Another option is to bypass Siebel Analytics Server security and rely on the security provided by issuing user-specific database logons and passwords when the Siebel Analytics Server submits queries to databases. The databases can then determine whether the query will be performed for the user.

The Siebel 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 Siebel Analytics Server 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 on connection pools, see [“Setting up Connection Pools” on page 91](#).

Bypass Siebel Analytics Server security by setting the authentication type in the NQSConfig.INI file:

```
AUTHENTICATION_TYPE = BYPASS_NQS;
```

See *Siebel Analytics Installation and Configuration Guide* for additional details.

Managing Query Execution Privileges

The Siebel 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 degree of query controls on users, no controls at all, or somewhere in between. The following lists 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 366](#).

- 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 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.
- 6 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 36](#) 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 36](#) 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 the Permissions tab.
- 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, click the 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 36](#) as a guide.

10 Click OK twice to return to the Security Manager dialog box.

Table 36. 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.
Warn	<ul style="list-style-type: none"> ■ Status Max Rows or Status Max Time. If the row limit is reached, a message will be logged in the NQServer.log file, and in the NQQuery.log file if logging is enabled for the user. The query will continue to run and the Siebel Analytics Server Administrator can use the information in the log entry to identify the query. If the logging is not enabled for a user, the user receives an error message. When Max Rows is exceeded the following message appears: <pre>The user request exceeded the maximum query governing rows from the database</pre> ■ Filter. This status is not available for filters.

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 Siebel Analytics Server supports various XML access modes, including access through the Siebel 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 389](#)
- [“Using the Siebel Analytics Server XML Gateway” on page 391](#)
- [“Using XML ODBC” on page 409](#)
- [“XML Examples” on page 411](#)

Locating the XML URL

The Siebel 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, `http://216.217.17.176/[DE0A48DE-1C3E-11D4-97C9-00105AA70303].XML`
- Dynamic XML generated from a server site. For example, `http://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:/htmldir/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:/htmldir/exam*.htm
d:/htmldir/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 402](#) 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 = valueof(session_variable1)&password = valueof(session_variable2)`. (This functionality also allows the Siebel 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 15, “Using Variables in the Analytics Server Repository.”](#)

The Siebel 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 Siebel Analytics Server needs to access any nonlocal files (network files or files on the Internet, for example), you need to run the Siebel 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 Siebel Analytics Server Runs”](#) on page 267.

Using the Siebel Analytics Server XML Gateway

Using the Siebel 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 Siebel 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. Siebel Systems currently supports 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 datatypes of the columns in the Physical layer, overriding the datatypes 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 datatype to other datatypes in the Business Model and Mapping layer of the Administration Tool, using the CAST operator.

At this time, the Siebel 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 [“Siebel Analytics Server XML Gateway Example” on page 394](#)).
- Element and attribute inheritance contained within the Microsoft XML schema.
- Element types of a mixed content model (such as XML elements which contain a mixture of elements and CDATA, such as `<p> hello Joe , how are you doing? </p>`).

NOTE: The Siebel 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 404](#).

To import XML data using the Siebel 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 Siebel Analytics Server XML Gateway supports all data sources described in the section [“Locating the XML URL” on page 389](#).

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. If you hold the Shift key down and click the Browse button, the Browse For Folder dialog box appears, which lets you browse and select a directory. For more information about variables, see [Chapter 15, “Using Variables in the Analytics Server 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 Siebel 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.

The Connection Pool dialog box appears.

- 9 Type a name and optional description for the connection on the General tab. See [“Setting up Connection Pools” on page 91](#) 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 Siebel Analytics Server XML Gateway allows users to specify a refresh interval for these data sources.

For information about the refresh interval for XML data sources, see [“About the Refresh Interval for XML Data Sources” on page 315](#) in Chapter 13, “Query Caching in Siebel Analytics Server.”

The default time-out interval for queries (URL loading time-out) is 15 minutes.

- 11 Click OK to complete the import.

- 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.

Siebel 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>
      </p4>
    </p2>
  </row>
</test>
```

```

        <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="eltOnly" order="many">
    <element type="row"/>
  </ElementType>
  <ElementType name="row" content="eltOnly" order="many">
    <element type="p1"/>
    <element type="p2"/>
  </ElementType>
  <ElementType name="p2" content="eltOnly" 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="eltOnly" 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>
      <QTY>2000</QTY>
    </ITEM>
  </ITEMS>
</CATEGORY>
</PRODUCTS>
</CUSTOMER>
<CUSTOMER>
  <CUST_ID>2</CUST_ID>
  <FIRST_NAME>Andrew</FIRST_NAME>
  <LAST_NAME>Carnegie</LAST_NAME>
  <ADDRESS>
    <ADD1>2955 Campus Dr.</ADD1>
    <ADD2>Ste. 300</ADD2>
    <CITY>San Mateo</CITY>
    <STATE>CA</STATE>
    <ZIP>94403</ZIP>
  </ADDRESS>
</CUSTOMER>
</PRODUCTS>
<CATEGORY>
  <CATEGORY_ID>CAT22</CATEGORY_ID>
  <CATEGORY_NAME>NAMEA1</CATEGORY_NAME>
  <ITEMS>
    <ITEM>
      <ITEM_ID>122</ITEM_ID>

```

```
        <NAME>DDDCOKE</NAME>
        <PRICE>11.50</PRICE>
        <QTY>2</QTY>
    </ITEM>
    <ITEM>
        <ITEM_ID>22</ITEM_ID>
        <NAME>PSPRITE</NAME>
        <PRICE>9.30</PRICE>
        <QTY>1978</QTY>
    </ITEM>
</ITEMS>
</CATEGORY>
<CATEGORY>
    <CATEGORY_ID>CAT24</CATEGORY_ID>
    <CATEGORY_NAME>NAMEA2</CATEGORY_NAME>
    <ITEMS>
        <ITEM>
            <ITEM_ID>19</ITEM_ID>
            <NAME>SOME COKE</NAME>
            <PRICE>1.58</PRICE>
            <QTY>3</QTY>
        </ITEM>
        <ITEM>
            <ITEM_ID>15</ITEM_ID>
            <NAME>DIET SPRITE</NAME>
            <PRICE>9.30</PRICE>
            <QTY>12000</QTY>
        </ITEM>
    </ITEMS>
</CATEGORY>
</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	// INVOICE/CUSTOMER/ADDRESS/ADD1
ADD2	// INVOICE/CUSTOMER/ADDRESS/ADD2

Using XML as a Data Source for Analytics

Using the Siebel Analytics Server XML Gateway

Column	Fully Qualified Name
CITY	// INVOICE/CUSTOMER/ADDRESS/CITY
STATE	// INVOICE/CUSTOMER/ADDRESS/STATE
ZIP	// INVOICE/CUSTOMER/ADDRESS/ZIP
CUST_ID	// INVOICE/CUSTOMER/CUST_ID
FIRST_NAME	// INVOICE/CUSTOMER/FIRST_NAME
LAST_NAME	// INVOICE/CUSTOMER/LAST_NAME
CATEGORY_ID	// INVOICE/CUSTOMER/PRODUCTS/CATEGORY/CATEGORY_ID
CATEGORY_NAME	// INVOICE/CUSTOMER/PRODUCTS/CATEGORY/ CATEGORY_NAME
ITEM_ID	// INVOICE/CUSTOMER/PRODUCTS/CATEGORY/ITEMS/ITEM/ ITEM_ID
NAME	// INVOICE/CUSTOMER/PRODUCTS/CATEGORY/ITEMS/ITEM/ NAME
PRICE	// INVOICE/CUSTOMER/PRODUCTS/CATEGORY/ITEMS/ITEM/ PRICE
QTY	// INVOICE/CUSTOMER/PRODUCTS/CATEGORY/ITEMS/ITEM/ QTY

Only tags with values are extracted as columns. An XML query generates fully qualified tag names, which helps to 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      PRICE   QTY   NAME
-----
Andrew       Carnegie        1.58    3     SOME COKE
Andrew       Carnegie       11.50    2     DDDCOKE
Andrew       Carnegie        9.30   12000  DIET SPRITE
Andrew       Carnegie        9.30   1978   PSprite
Margar       Leverling       0.38   20099  SPRITE33
Margar       Leverling        1.0     3     COLA
```


Margar	Leverling	30.50	20033	COKE33
Margar	Leverling	8.30	123	MY SPRITE
Nancy	Fuller	0.30		SPRITE
Nancy	Fuller	0.50	2000	
Nancy	Fuller	1.50	3000	ACoke
Nancy	Fuller	3.30	2000	SOME SPRITE

Row count: 12

Accessing HTML Tables

The Siebel 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 Siebel 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 Siebel 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 cellpadding=2 cellspacing=0>
      <tr>
        <th colspan=1>Transaction</th>
        <th colspan=2>Measurements</th>
      </tr>
      <tr>
        <th>Quality</th>
        <th>Count</th>
        <th>Percent</th>
      </tr>
      <tr>
        <td>Failed</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>Warning</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>

```

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 Siebel 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 411](#).

Using the Data Mining Adapter

The Data Mining Adapter is an extension of the Siebel 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 162](#).

The Data Mining Adapter has two modes of operation:

- **In Process.** 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, which retrieves records one row at a time. The query results are returned to the XML gateway through an API parameter.
- **Out of Process.** 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.

The In-Process 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 the file IterativeGatewayDll.h for the definition of the data type and structures used in this API.

```

extern "C" ITERATIVEGATEWAYDLL_API SiebelAnalyticIterativeExecutionStatus(
    /* [in] */ const wchar_t * modelId

    /* [in] */ const int inputColumnCount
    /* [in] */ const SiebelAnalyticColumnMetaInfo pInputColumnMetaInfoArray
    /* [in] */ const uint8 inputColumnValueBuffer

    /* [in] */ const int OutputColumnCount, //actual
count of columns returned
    /* [in/out] */ SiebelAnalyticColumnMetaInfo pOutputColumnMetaInfoArray
    /* [out] */ uint8 outputColumnValueBuffer

```

Table 37 provides a description of the API elements.

Table 37. 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, which is included with Siebel 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.

Table 37. API Elements

Element	Description
pOutputColumnMetaInfoArray	An array of meta column information for the output column. SiebelAnalyticColumnMetaInfo is declared in the public header file IterativeGatewayDll.h, which is included with Siebel 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 Analytics installation folder. 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)
- xlC50.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

When you use the out-of-process mode, that is, 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, \$(NQ_OUT_TEMP_FILE) to which the executable outputs the result line. When the Data Mining Adapter invokes the executable, the placeholder \$(NQ_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) $(NQ_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 Using the Administration Tool, create a database and select XML Server as the database type.

For information about creating a database, see [“Creating a Database Object in the Physical Layer” on page 88](#).

- 2 Configure the connection pool:
 - a Right-click the database you created in Step 1, and then select New Object > Connection Pool.
 - b Type a name for the connection pool.
 - c Select XML as the call interface.
 - d Type a data source name, and then click OK.

NOTE: Do not type information into any field in the XML tab of the Connection Pool dialog box. The empty fields indicate to the Siebel Analytics Server that the Data Mining Adapter functionality will be invoked.

- 3 Right-click the database you created in Step 1, and then select New Object > Table.

The Physical Table dialog box appears. In the XML tab you specify which mode you want to use: in process or out of process. For information about these modes, see [“Using the Data Mining Adapter” on page 404](#).

- 4 In the XML tab of the Physical Table dialog box, do one of the following:
 - Select the Executable radio button, type the path to the executable file in the Search Utility field, and specify the delimiter for the output values.
 - Select the DLL radio button 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:

```
"D:\SiebelAnalytics\Bin\Data Mining DLL\ExternalGatewayDLL.dll"
```

All characters appearing after the DLL path are passed down 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:

```
"D:\SiebelAnalytics\Bin\Data Mining DLL\ExternalGatewayDLL.dll  
VALUEOF(Model1) VALUEOF(Model2)"
```

Using XML ODBC

Using the XML ODBC database type, you can access XML data sources through an ODBC interface. The datatypes of the XML elements representing physical columns in physical tables are derived from the datatypes of the XML elements as defined in the XML schema. In the absence of a proper XML schema, the default datatype of string is used. Datatype 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 datatype 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 Import the ODBC DSNs into the repository, making sure you select the Synonyms option in the Import dialog box, and then click OK.

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 xmlns:s='uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882'
      xmlns:dt='uuid:C2F41010-65B3-11d1-A29F-00AA00C14882'
      xmlns:rs='urn:schemas-microsoft-com:rowset'
      xmlns:z='#RowsetSchema'>
  <s:Schema id='RowsetSchema'>
    <s:ElementType name='row' content='eltOnly'
      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='i2' 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 Siebel 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="eltOnly" order="many">
    <AttributeType name="height" dt:type="int" />
    <element type="row"/>
  </ElementType>
  <ElementType name="row" content="eltOnly" order="many">
    <element type="p1"/>
    <element type="p2"/>
  </ElementType>
  <ElementType name="p2" content="eltOnly" 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"/>
    <element type="p5"/>
  </ElementType>
  <ElementType name="p4" content="eltOnly" order="many">
    <element type="p3"/>
    <element type="p6"/>
    <element type="p7"/>
  </ElementType>
  <ElementType name="test0" content="eltOnly" order="many">
    <element type="row"/>
  </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>

```

84.xml

```
===84.xml===

<?xml version="1.0"?>
<test0 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>
</test0>
```

Island2.htm

```
===island2.htm===  
  
<HTML>  
  <HEAD>  
<TITLE>HTML Document with Data Island</TITLE>  
</HEAD>  
  <BODY>  
<p>This is an example of an XML data island in I.E. 5</p>  
  <XML ID="12345">  
    <test>  
      <row>  
        <field1>00</field1>  
        <field2>01</field2>  
      </row>  
      <row>  
        <field1>10</field1>  
        <field2>11</field2>  
      </row>  
      <row>  
        <field1>20</field1>  
        <field2>21</field2>  
      </row>  
    </test>  
  </XML>  
<p>End of first example.</p>  
<XML ID="12346">  
  <test>  
    <row>  
      <field11>00</field11>  
      <field12>01</field12>  
    </row>  
    <row>  
      <field11>10</field11>  
      <field12>11</field12>  
    </row>  
    <row>  
      <field11>20</field11>  
      <field12>21</field12>  
    </row>  
  </test>  
</XML>  
<p>End of second example.</p>  
</BODY>  
</HTML>
```


The Siebel 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 418](#)
- [“SELECT Usage Notes” on page 419](#)
- [“SELECT List Syntax” on page 420](#)
- [“Rules for Queries with Aggregate Functions” on page 421](#)

NOTE: The syntax descriptions throughout this guide are not comprehensive. They cover only basic syntax and features unique to the Siebel 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 Siebel 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 Siebel 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 420 .
<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 420 .
<i>search_condition</i>	Specifies any combination of conditions to form a conditional test. See “WHERE Clause Syntax” on page 420 .
<i>column</i>	A column (or alias) belonging to a table defined in the data source.

SELECT Usage Notes

The Siebel 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 Siebel 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 Siebel 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 Siebel 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 Siebel 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 Siebel 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 Siebel Analytics Server determines the proper tables and the proper join specifications based on the columns the query asks for and the configuration of the Siebel Analytics Server repository.

WHERE Clause Syntax

The Siebel 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 Siebel Analytics Server repository. Any join conditions specified in the WHERE clause are ignored.

The Siebel 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 Siebel 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 Siebel 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 Siebel Analytics Server, see [“Rules for Queries with Aggregate Functions” on page 421](#).

ORDER BY Clause Syntax

The Siebel 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 Siebel Analytics Server simplifies the SQL needed to craft aggregate queries. This section outlines the rules that the Siebel 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 Siebel 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, which in this case is the sales of Coke plus the sales of Pepsi.

If you add a column to the query requesting the COUNT of *revenue*, the Siebel Analytics Server calculates the number of records used to calculate the results for each group, which is a year in this case, as follows:

```
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 Siebel 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, which in this case 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 Siebel 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 Siebel 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”](#)
- [“Running Aggregate Functions” on page 439](#)
- [“String Functions” on page 445](#)
- [“Math Functions” on page 453](#)
- [“Calendar Date/Time Functions” on page 460](#)
- [“Conversion Functions” on page 471](#)
- [“System Functions” on page 474](#)
- [“Expressing Literals” on page 474](#)

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_expression)
```

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 (DISTINCT n_expression)
```

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_expression, n)
```

where:

n_expression Any expression that evaluates to a numerical value.

n_expression 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 numeric 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 n days.

You should not use the FIRST function as the first dimension-specific aggregate rule; doing so can cause poor performance because it might cause queries to bring back large numbers of rows for processing in the Siebel Analytics Server.

Syntax:

FIRST (*n_expression*)

where:

n_expression Any expression that evaluates to a numerical value.

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 numeric 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 *n* days.

You should not use the LAST function as the first dimension-specific aggregate rule; doing so can cause poor performance because it might cause queries to bring back large numbers of rows for processing in the Siebel Analytics Server.

Syntax:

```
LAST (n_expression)
```

where:

n_expression Any expression that evaluates to a numerical value.

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 425](#).

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:

```
MEDIAN (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 425](#).

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 425](#).

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 Siebel Analytics Server percentile function, which 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 425](#).

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 425](#).

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 425](#).

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 425](#).

SumDistinct

Calculates the sum obtained by adding all of the distinct values satisfying the numeric expression argument.

Syntax:

```
SUM(DISTINCT n_expression)
```

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_expression, n)
```

where:

n_expression Any expression that evaluates to a numerical value.

n_expression 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 425](#).

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 Siebel 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:

<i>n_expression</i>	Any expression that evaluates to a numerical value.
<i>n</i>	Any positive integer. Represents the average of the last <i>n</i> 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 425](#).

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 nth 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 425](#).

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
APRIL	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 425](#).

Syntax:

```
RSUM (n_expression)
```

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 RUNNING_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
APRIL	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 425](#).

Syntax:

```
RCOUNT (Expr)
```

Where:

Expr An expression of any datatype.

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 datatype 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 425](#).

Syntax:

```
RMAX (expression)
```

Where:

expression An expression of any datatype. The datatype 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 datatype 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 425](#).

Syntax:

```
RMIN (expression)
```

Where:

expression An expression of any datatype. The datatype 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, which is 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_expression)
```

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_expression1, character_expression2)
```

where:

character_expression Expressions that evaluate to character strings.

Form 2 Syntax:

```
CONCAT (string_expression1 || string_expression2 || ...  
string_expressionxx)
```

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, which is then 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_expression, n, m, character_expression)
```

where:

character_expression Any expression that evaluates to a character string.

n Any positive integer representing the number of characters from the start of the first string where a portion of the second string is inserted.

m 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:

character_expression Any expression that evaluates to a character string.

n 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:

character_expression 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:

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.

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_expression1, character_expression2, 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_expression)
```

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_expression)
```

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:

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.

Upper

Converts a character string to uppercase.

Syntax:

```
UPPER (character_expression)
```

where:

character_expression 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:

n_expression 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:

n_expression Any expression that evaluates to a numerical value.
n 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:

n_expression Any expression that evaluates to a numerical value.

Sin

Calculates the sine of a numerical expression.

Syntax:

```
SIN (n_expression)
```

where:

n_expression 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:

n_expression 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 Siebel Analytics Server is running.

Syntax:

```
CURRENT_DATE
```

Current_Time

Returns the current time. The time is determined by the system in which the Siebel 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 Siebel 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:

```
    HOUR (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:

<i>interval</i>	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
<i>integer_expression</i>	Any expression that evaluates to an integer.
<i>timestamp_expression</i>	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_expression The timestamp used in the function.

1

timestamp_expression The first timestamp used in the function.

2

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 Siebel Analytics Server administrator has configured the start of a new week in the `NQSConfig.INI` file using the parameter `FIRST_DAY_OF_THE_WEEK` (which 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 either a numeric value or a null value to another data type.

Syntax:

```
CAST (expression|NULL AS datatype)
```

For example, you can cast a `customer_name` (a data type of `Char` or `Varchar`) or `birthdate` (a `datetime` literal). The supported data types to which the value can be changed are the following:

```
CHARACTER, VARCHAR, INTEGER, FLOAT, SMALLINT, DOUBLE PRECISION,  
DATE, TIME, TIMESTAMP, BIT, BIT VARYING
```

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.

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 the Administration Tool. You can also use the VALUEOF function when you edit the SQL for a request from the Advanced tab in Siebel 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 Siebel Analytics repository to which you are logged in.

Syntax:

```
USER ( )
```

Database

Returns the name of the Siebel 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.

Usage Tracking Data Descriptions and Using the Log File Method

A

The Siebel Analytics Server supports the collection of usage tracking data. When usage tracking is enabled, the Siebel 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 information about administering usage tracking, see [“Administering Usage Tracking” on page 277](#).

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 Analytics applications, this is the name used in the Siebel 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 Systems provides the following sample JavaScript files located in the SiebelAnalytics\Scripts\Common subdirectory:

- sblAcctLoaderMSSQL for SQL Server
- sblAcctLoaderOCL.js for Oracle
- sblAcctLoaderADO.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 477](#).

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 38](#) describes each column in the usage tracking table.

Table 38. 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.

Table 38. Usage Tracking Data

Column	Description
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 Siebel 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 Siebel Analytics Server waited for back-end physical databases on behalf of a logical query.
CUM_NUM_DB_ROW	The total number of rows returned by the back-end databases.
CACHE_IND_FLG	Y indicates a cache hit for the query, N indicates a cache miss.
QUERY_SRC_CD	The source of the request, e.g., Drill or Report.

Table 38. Usage Tracking Data

Column	Description
SAW_SRC_PATH	The path name within the Analytics web catalog for the request.
SAW_DASHBOARD	The name of the dashboard. Null if the query was not submitted via an Analytics dashboard.

Pharma components for Disconnected Analytics are installed during the Siebel Analytics installation in the \SiebelAnalyticsData\Disconnected\Pharma directory.

The following is a list of components that have been tailored for Pharma Analytics and their locations:

- **PharmaDisconnect.rpd.** The Pharma Analytics repository is tailored for Pharma 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 Analytics. It is located in the \SiebelAnalyticsData\Disconnected\Pharma\Application directory.
- **Pharma.XML.** Application configuration file tailored for Pharma Analytics. It is located in the \SiebelAnalyticsData\Disconnected\Pharma directory.

This appendix contains the reference material about the following topics for Pharma Analytics administrators and system administrators.

- [Sourcing Reports for Pharma Analytics](#)
- [Data Sets for Pharma Analytics Data on page 483](#)

Sourcing Reports for Pharma Analytics

The sourcing reports for Pharma Analytics issue logical SQL to the Siebel Analytics server to extract the data used to populate the local disconnected client schema.

Table 39. Sourcing Reports for Pharma 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
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

Table 39. Sourcing Reports for Pharma Analytics

Subject Area	Sourcing Report Name	Target Table
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 Analytics Data

Please refer to the Pharma configuration file (Pharma.XML) for details.

Table 40.

Dataset 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

Table 40.

Dataset Name	Sourcing Report Name	Target Table
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

Table 40.

Dataset Name	Sourcing Report Name	Target Table
SyndicateData	Prescription Sales Fact	W_SYND_RX_F
SyndicateData	Prescription Sales Aggregate	W_SYND_RX_N_A

Index

A

- Abs math function, about 453
- Acos math function, about 453
- Administration Tool
 - See also* Browse dialog box; logical business models; repository mode
 - about 32
 - Cache Manager, changing column order 69
 - Cache Manager, selecting columns to display 69
 - Edit menu, described 53
 - File menu, described 53
 - Help menu, described 54
 - icons and symbols (table) 56
 - join diagrams, setting default window size 69
 - keyboard shortcuts (table) 55
 - main window, repository parts described 51
 - Manage menu, described and functions (table) 53
 - preferences, setting general preferences 65
 - repository components 75
 - repository objects, adding or editing 70
 - repository objects, setting permissions 70
 - repository objects, specifying appearance in alphabetical order 68
 - row counts, about updating for tables and columns 71
 - scrolling speed, setting 69
 - Siebel Analytics Server, using to shut down 271
 - toolbar functions 54
 - Tools menu, described 54
 - View menu, described 53
 - Window menu, described 54
- aggregate expression builder dialog
 - See* Expression Builder dialog boxes
- aggregate fact table
 - creating and example, about 250
 - sources, creating for each logical table 252
- aggregate functions
 - about 430
 - aggregate queries, about 421
 - alternative syntax 426
 - Avg aggregate function, about 430
 - AvgDistinct, calculating average mean 431
 - baseline columns, computing aggregates 421
 - BottomN, about ranking the lowest n values 431
 - Count (*) (CountStar), about counting number of rows 432
 - Count, about calculating the number of rows 431
 - CountDistinct, about adding distinct process to COUNT 432
 - display function, reset behavior 425
 - First, about selecting first returned value 432
 - GroupByColumn, about specifying logical columns 433
 - GroupByLevel, about setting up aggregate navigation 433

- LAST, about selecting last returned value 434
- Max, about calculating maximum value 434
- measure columns, computing aggregates 423
- Median, about calculating the median value of rows 434
- Min, about calculating the minimum value 435
- NTILE, about determining the rank of a value 435
- Percentile, about calculating a percent rank 436
- Rank, about calculating the rank for each value 436
- StdDev, about returning standard deviation 437
- Sum, about calculating 437
- SumDistinct, about calculating sum by adding distinct values 438
- TopN, about ranking the highest n values 438
- aggregate navigation, setting up
 - See also* aggregate tables; aggregate table fragments
- aggregate levels, specifying for each source 250
- aggregate table definitions, about and navigation 43
- aggregate table fragments, about and example 259
- aggregate tables, about 249
- aggregated fact data, creating dimension sources for each level 250
- aggregated fact data, creating sources for each logical table 252
- fragment content, specifying 253
- WHERE clause filter, about and example 250
- aggregate table fragments
 - about and example 259
 - about configuring a repository to use fragments 260
 - aggregate table content, specifying 261
 - physical joins for virtual table, constructing 264
 - physical layer table, defining with a Select Statement 261
 - physical layer table, example 262
 - SQL virtual table content, creating 263
- aggregate tables
 - See also* aggregate table fragments
 - about and navigation 249
 - aggregate table definitions, about and navigation 43
 - performance, about using to improve 287
- aliases
 - Alias tab, using 174
 - synchronizing 234
- analyzing the cache
 - See* Cache Manager
- application environment, about using metadata repositories to store information 32
- architectural components
 - Administration Tool, about 32
 - diagram 31
 - metadata repositories, about 32
 - multithreaded architecture 32
 - Web client tools and ODBC-compliant tool, about 33
 - Web Server and Intelligence Dashboards, about 33
- ASCII string function, about 445
- Asin math function, about 453
- Ask DBMS button, using to change Feature table entries 91
- Atan math function, about 454
- Atan2 math function, about 454
- audience for guide 15
- authentication cache, about disabling 372
- authentication options
 - See also* security; groups, working with

- authentication, about 374
 - authentication, order of 383
 - database authentication, about using and procedure 381
 - external table authentication, about 378
 - external table authentication, setting up 378
 - LDAP authentication, about 376
 - LDAP authentication, setting up 376
 - logging level, setting 377
 - operating system authentication, about 375
 - operating system authentication, configuring (procedure) 375
 - password, changing 382
 - security, bypassing 383
 - Siebel Analytics Server user IDs and passwords, about and storage of 382
 - Siebel Server internal authentication, about 382
 - USER session system variable, defining for LDAP authentication 377
 - Avg aggregate function, about 430
 - AvgDistinct aggregate function, about 431
- B**
- baseline column
 - behavior with aggregate functions 421
 - example 422
 - Between SQL logical operator, about 427
 - Bit_Length string function, about 445
 - BottomN aggregate function, about 431
 - bridge tables, using to model many-to-many relationships 46
 - Browse dialog box
 - about using 72
 - querying for an object 73
 - selecting an object in 73
 - synchronizing an object in the query results list with the tree control list 73
 - buffer size, configuring 294
 - Business Model and Mapping layer
 - about creating and maintaining 127
 - Business Model and Mapping layer, working in
 - See also* business models; logical table joins
 - business model, about working with 128
 - business model, creating (procedure) 128
 - logical columns, creating 134
 - logical table source 137
 - logical table sources (mappings), working with 137
 - logical tables, about working with 129
 - measure, associating with a level in dimension 136
 - measure, removing the association 137
 - physical to logical mapping, defining 139
 - repository, about setting up and working in 76
 - table content definitions, creating 141
 - business models
 - See also* Business Model and Mapping layer, working in; presentation catalogs
 - consistency check, about passing 63
 - copy business model with presentation catalog utility 236
 - diagram, about using to create joins 157
 - diagram, displaying 160
 - repository, checking consistency within a 63
 - understanding 35
- C**
- cache
 - behavior in offline mode 310
 - behavior in online mode 310
 - disabling 295
 - enabling 294
 - purging 313

- purging when switching between repositories 310
- cache authentication, about disabling 372
- cache hits
 - description of 298
 - information about 298
- cache information, viewing 356
- Cache Manager
 - about and opening 311
 - column order, changing 69
 - columns, selecting columns to display 69
 - global cache information, displaying 312
 - option settings (table) 311
 - purging cache 313
 - view and save SQL call 311
- cache persistence time setting 296
- cache storage
 - cache data storage directories 293
 - cache entry, controlling max number of rows 294
 - cache metadata file, about 293
 - query caching, disabling query caching 295
 - query caching, enabling query caching 294
- Cache, Manage menu option, described 53
- cache, monitoring and managing
 - disabling caching for the system, about and advantages 295
 - event polling tables, configuring 296
 - physical tables, about caching and persistence timing 296
- Calculation Wizard, about 66
- calendar date/time functions
 - Current_Date, about returning 460
 - Current_Time, about 461
 - Current_TimeStamp, about 461
 - Day_Of_Quarter, about 461
 - DayName, about 462
 - DayOfMonth, about 462
 - DayOfWeek, about 462
 - DayOfYear, about 463
 - Hour, about 463
 - Minute, about 463
 - Month, about 463
 - Month_Of_Quarter, about 464
 - MonthName, about 464
 - Now, about 464
 - Quarter_Of_Year, about 465
 - Second, about 465
 - TimestampAdd, about 465
 - TimeStampDiff, about 468
 - Week_Of_Quarter, about 470
 - Week_Of_Year, about 470
 - Year, about 471
- Cast conversion function, about 471
- catalog folders
 - creating or editing a catalog 115
 - Dynamic Name tab, sorting entries 116
 - Dynamic Name tab, specifying the session variable to use 116
 - Dynamic Name tab, unassigning a session variable 116
- Ceiling math function, about 454
- Char string function, about 445
- Char_Length string function, about 446
- character literals, about 474
- checking in changes
 - changes, making available and saving to disk 64
 - Check In Changes dialog box, about using and tasks 64
- checkpoints
 - described 283
- client tools, connectivity to 320
- cluster information, viewing
 - cache information, viewing 356
 - Cache view, about and columns described (table) 356
 - Request window (Session view), columns described (table) 357
 - Session window (Session view), columns described (table) 357

- Session window, columns described (table) 357
- Status view, about and columns described (table) 354
- Cluster Manager
 - See also* Cluster Server feature
 - accessing (procedure) 353
 - cache information, viewing 356
 - Cache view, about and columns described (table) 356
 - graphical user interface, described 353
 - managing clustered servers (procedure) 359
 - managing clustered servers, actions available 359
 - performance considerations 360
 - refreshing, setting and procedure 353
 - Session view, Request window (Session view) columns described (table) 357
 - Session view, Session and Request window, described 357
 - Session view, Session window columns described (table) 357
 - Status view, about and columns described (table) 354
 - stopped or offline, note about starting 353
 - using, about 353
- Cluster Server feature
 - See also* Cluster Manager
 - about 345
 - cache information, viewing 356
 - Cluster Controller and Analytics Server, starting from the Command window 350
 - Cluster Controller and Analytics Server, starting manually in Windows 350
 - installing 347
 - managing clustered servers (procedure) 359
 - managing clustered servers, actions available 359
 - master server, about 346
 - NSClusterConfig.INI file, about copying to Config directory 349
 - parameters, setting in the NQClusterConfig.INI file 348
 - parameters, setting in the NQSConfig.INI file 347
 - performance considerations 360
 - primary Cluster Controller, about 345
 - Repository Publishing Directory, about 346
 - secondary Cluster Controller, about 346
 - Siebel Analytics ODBC data source name, about configuring 349
 - startup process overview 351
 - stopped or offline, note about starting 353
- Cluster, Manage menu option, described 53
- column mapping
 - logical columns to physical columns, mapping 139
 - removing column mapping 140
- Command window, stating Cluster Controller and Analytics Server 350
- complex joins
 - about 120
 - logical complex joins, about and complex joins 157
 - logical complex joins, creating 159
- Concat string function, about 446
- conditional expressions
 - CASE (if), about and syntax 428
 - CASE (Switch), about and syntax 427
- connection pool
 - about 76
 - creating and configuring, about 91
 - creating or editing 93
 - dialog box, fields described 94
- connectivity
 - client tools and data sources, connectivity to 320
 - metadata, importing 320

- ODBC data source names,
 - configuring 317
 - query and reporting tools, about
 - using 321
 - consistency check
 - about passing and example 63
 - business model within a repository,
 - checking consistency of 63
 - repository, checking for consistency 63
 - conversion functions
 - Cast, about changing the data type to
 - another data type 471
 - IfNull, about testing if an expression
 - evaluates to a null value 472
 - VALUEOF(), about using the function in
 - an expression builder 472
 - copy business model with presentation
 - catalog utility, about and
 - procedure 236
 - Cos math function, about 455
 - Cot math function, about 455
 - Count (*)/CountStar aggregate function,
 - about 432
 - Count aggregate function, about 431
 - CountDistinct aggregate function,
 - about 432
 - Current_Date calendar date/time function,
 - about 460
 - Current_Time calendar date/time function,
 - about 461
 - Current_TimeStamp calendar date/time
 - function, about 461
- D**
- Data Mining Adapter
 - See also* XML Gateway; XML ODBC
 - database type; XML, using as a data
 - source
 - configuring (procedure) 408
 - In-Process Data Mining Adapter API,
 - about 404
 - In-Process Data Mining Adapter API,
 - sample implementation 406
 - In-Process Data Mining Adapter API,
 - using ValueOf() expression 407
 - operation modes 404
 - data modeling
 - business model, understanding 35
 - logical business models, about 44
 - objectives of 35
 - physical database model,
 - understanding 37
 - data sources
 - connectivity to 320
 - heterogeneous, about 44
 - data transformations, about 28
 - data warehouse, list of construction
 - tasks 27
 - DATA_STORAGE_PATHS parameter, using
 - to specify query cache storage 293
 - database hints
 - See also* databases
 - database objects that accept objects
 - (table) 124
 - hints, creating (procedure) 125
 - index hint, about 124
 - Leading hint, about 125
 - performance considerations 125
 - SQL comment markers, about
 - entering 126
 - usage examples 124
 - using, about 123
 - database object
 - creating or editing in the Physical
 - layer 88
 - database hints, database objects that
 - accept hints (table) 124
 - ODBC type, about assigning if database
 - type undetermined 88
 - Database system function, about 474
 - database type, restoring default entries
 - for 91
 - databases
 - See also* database hints
 - authentication, about using and
 - procedure 381

- configuring, tuning, and indexing, importance 288
 - database hints, about using 123
 - symbol, described 56
- datetime literals, about 474
- Day_Of_Quarter calendar date/time function, about 461
- DayName calendar date/time function, about 462
- DayOfMonth calendar date/time function, about 462
- DayOfWeek calendar date/time function, about 462
- DayOfYear calendar date/time function, about 463
- DCOM, changed client/server communication method 319
- decimal literal, about 475
- default client/server communication method, changed from DCOM to TCP/IP 319
- Default initializer column, about value 342
- Degrees math function, about 455
- deleting
 - alias 174
 - column mapping 140
 - initialization block 343
 - measure, removing the association 137
 - presentation tables 170
 - presentation tables column 173
 - table as a logical table source 139
- description of 298
- Diagnosis Records table, about limiting the number of records 48
- dimensional hierarchies
 - factual measures, about 46
 - sample hierarchy rollups, about and diagram 45
 - star and snowflake models, about 46
- dimensional hierarchy
 - grand total levels, about 145
 - grand total levels, example 151
 - grand total, example 151
 - level attributes, about 145
 - level-based measure calculations, setting up 149
 - level keys, about 146
 - level-based measure calculations, about 149
 - level-based measure calculations, example 149
 - levels, about creating 144
- dimensional level
 - general properties, defining 147
 - primary key, adding 148
- dimensional models
 - See also* individual dimensional entries about 37
 - bridge tables, using to model many-to-many relationships 46
 - sample hierarchy rollups, about and diagram 45
 - single table model, about and creating 49
 - understanding 44
- dimensional schemas
 - about and advantages 38
 - start schema (diagram) 39
- dimensions
 - about 143
 - about cubes from a multidimensional data source 143
 - creating (procedure) 144
 - creating and administering 143
 - hierarchies, about 143
 - note, about including the key column in the dimension 144
- dimensions, defined and example 36
- dimension-specific aggregation rules
 - columns, specifying for 154
 - setting up, about 153
- dirty data, about 28
- Disconnected Analytics
 - Application Manager utility 203
- display functions

- example 425
 - reset behavior 425
 - DISPLAYNAME system session variable,
 - about 330
 - dragging and dropping
 - logical tables 130
 - driving table
 - about specifying 162
 - caution, about specifying when creating a Business Model Diagram 161
 - caution, about specifying when creating a logical complex join 160
 - caution, about specifying when creating a logical foreign key 159
 - caution, specifying and query optimization 163
 - controlling and tuning performance, about 162
 - logical joins, specifying (procedure) 162
 - DSN connection
 - See Siebel Analytics Server
 - Dynamic Name tab
 - entries, sorting 116
 - session variable, specifying 116
 - session variable, unassigning 116
 - dynamic repository variables
 - See also initialization blocks
 - about 327
 - example 327
 - initializing 333
- E**
- Edit menu, described 53
 - EMAIL system session variable, about 331
 - entity-relationship (E-R) models
 - about 37
 - queries that perform historical analysis, performance of 37
 - environment diagram, high-level 25
 - event polling table
 - cache event processing, about 302
 - configuring 296
 - CREATE TABLE statements, sample event polling table 306
 - making the polling table active 307
 - overview 296
 - physical databases, setting up 304
 - populating, about 309
 - repository. making changes to 310
 - Siebel Analytics Event Table utility, identifying using 236
 - troubleshooting 309
 - using 309
 - Execute UDML utility
 - about and starting 237
 - Exp math function, about 456
 - Export logical keys option, about using with parameterized SQL queries 169
 - Expression Builder dialog boxes
 - about using 241
 - accessing 242
 - Aggregate Content folder, about 245
 - Constraints folder, about 245
 - Dimensions folder, about 245
 - example (diagram) 243
 - expression, building (procedure) 248
 - Expressions folder, about 245
 - Functions folder, about 245
 - Logical Tables folder, about 245
 - navigating within the expression builder 247
 - Operators folder, about 245
 - Repository Variables folder, about 246
 - Session Variables folder, about 246
 - setting up example 246
 - toolbar (table) 244
 - Types folder, about 245
 - expression literals
 - character literals, about and expressing 474
 - datetime literals, about and formats 474
 - decimal, about and expressing 475
 - floating point, about and expressing 475
 - integers, about and expressing 475
 - numeric literals, about 475

- Extensible Markup Language
 - See* XML, using as a data source
- external table authentication
 - about 378
 - setting up 378
- Externalize Strings utility, about and starting 237
- F**
- fact table, about 40
- facts (factual measures), defined and aggregation rule example 36
- factual hierarchies, about dimensional hierarchies 46
- Feature table entries, changing using Ask DBMS 91
- File menu, described 53
- filter
 - See also* query execution privileges
 - complex filter, about constructing 185
 - constructing a filter to view all databases references in a business model 184
 - constructing a filter to view all Presentation layer columns mapped to a logical column 185
 - query results, constructing a filter for 184
- First aggregate function, about 432
- floating point literal, about 475
- Floor math function, about 456
- folder symbol, described 56
- foreign keys
 - Foreign Keys tab, using to specify a driving table 162
 - logical foreign key, creating 158
 - note, about importing from an Oracle database 83
 - primary key, relationship with 119
- fragmentation content
 - about 253
 - multicolumn content descriptions example 254
 - parallel content descriptions example 255
 - single column range-based predicates example 253
 - single column value-based predicates example 253
 - unbalanced parallel content descriptions example 258
- fragmented data, about and example 118
- FROM clause syntax, about 420
- functions
 - See* aggregate functions
- G**
- grand total dimension hierarchy
 - example 151
- grand total levels
 - about 145
 - example 151
- granularity, defined 40
- graphical user interface
 - See* Cache Manager
- GROUP BY clause
 - query behavior with and without 425
 - syntax, about 421
- GROUP system session variable, about 330
- GroupByColumn aggregate function, about 433
- GroupByLevel aggregate function, about specifying dimension levels 433
- groups, working with
 - See also* authentication options
 - about 365
 - groups, about creating and example 366
 - LDAP authentication, configuring 371
 - LDAP, about importing users and groups 371
 - member hierarchies, viewing in the Query Repository dialog box 370
 - member hierarchies, viewing in the Security Manager 370
 - predefined Administrators group, about 366

- privileges example (diagram) 368
- privileges, about granting and examples 366
- repository, adding a new group to 369
- repository, importing LDAP users and groups into 373
- user privileges and group privileges example (diagram) 367
- guide
 - audience for 15
- H**
- Help menu, described 54
- heterogeneous data sources, about 44
- HIDD_A_AGGREGATION 153
- HIDD_A_KEY 108
- HIDD_A_PROJECT 192
- HIDD_PROJECTS 192
- hierarchies
 - about 143
- hierarchy, defined and example 36
- historical time comparison, about and starting Times Series Wizard 234
- Hour calendar date/time function, about 463
- HTML tables
 - example 402
 - XML Gateway, accessing by 402
- I**
- icons, described (table) 56
- IfNull conversion function, about 472
- importing
 - metadata 320
 - metadata, about connecting using an ODBC connection 84
 - users and groups using LDAP 371
 - XML data using ODBC 409
- In SQL logical operator, about 427
- inconsistencies, about checking repository or business model for
 - consistencies 63
- indexing, about index hint instructions 124
- INI file
 - NQClusterConfig.INI file, setting parameters for Cluster Server feature 348
 - NQSConfig.INI file, collecting information about queries 283
 - NQSConfig.INI file, tuning parameters 287
 - NSQConfig.INI file, adding an entry 177
- Initialization Block dialog box
 - accessing (procedure) 341
 - using, about 341
- initialization blocks
 - about 333
 - block, linking with a variable not already refreshed 342
 - block, removing variable's association with 342
 - blocks, removing 343
 - caution, about opening Initialization Block dialog box in online mode 333
 - Default initializer column, about value 342
 - dynamic repository variables, initializing 333
 - initialization block execution order, setting 343
 - new variable, adding to refresh with block 341
 - note, about number of columns different from number retrieved 341
 - session variables, creating or editing (procedure) 336
 - session variables, initializing 334
 - Siebel Analytics Server, re-initializing when it starts 342
 - variable refreshed by this block, editing variable, reordering 341
- In-Process Data Mining Adapter API
 - about 404
 - column values, specifying 407

- configuring (procedure) 408
 - sample implementation 406
 - ValueOf() expressions, using 407
- Insert string function, about 447
- integers literals, about 475
- Is Null SQL logical operator, about 427
- J**
- Jobs, Manage menu option, described 53
- join diagrams, setting default window size 69
- Joins Manager
 - joins, about using to create 157
- Joins, Manage menu option, described 54
- K**
- key symbol, described 57
- keyboard shortcuts (table) 55
- L**
- Last aggregate function, about 434
- LDAP
 - See* Lightweight Directory Access Protocol (LDAP)
- Leading hint, about 125
- Left string function, about 447
- Length string function, about 448
- level attributes
 - about 145
- level keys
 - about 146
- level-based measure calculations
 - about 149
 - example 149
 - setting up 149
- level-based measure calculations, about 149
- levels
 - general properties, defining 147
 - hierarchy, about 145
 - primary key for, adding 148
- levels, working with
 - grand total levels, example 151
 - level-based measure calculations, setting up 149
- Lightweight Directory Access Protocol (LDAP)
 - authentication, about 376
 - authentication, setting up 376
 - LDAP authentication, configuring 371
 - logging level, setting 377
 - passwords and storage, about 382
 - repository, importing LDAP users and groups into 373
 - USER session system variable, defining for LDAP authentication 377
 - users and groups, about using to import 371
- Like SQL logical operator, about 427
- literals, expressing (list of) 474
- Load all objects option, about selecting 62
- Locate string function, about 448
- LocateN string function, about 448
- log files
 - See also* query log, administering; usage tracking, administering
 - NQServer.log and NQCluster.log, about opening and examining 350
 - See* log files
- Log math function, about 456
- log viewer utility
 - log records, interpreting 276
 - running (procedure) 275
 - using, about 274
- Log10 math function, about 456
- logging levels
 - individual users, about enabling for 273
 - levels described (table) 273
 - log viewer utility, interpreting the log records 276
 - log viewer utility, using 274
 - user's logging levels
 - disabling 274
 - setting 274
- logical business models

- about 44
 - bridge tables, using to model many-to-many relationships 46
 - dimensional models, understanding 44
 - logical tables and columns and
 - heterogeneous data sources 44
 - single table model, about and
 - creating 49
 - Logical Column dialog box
 - measure, associating with a level in a dimension 136
 - measure, removing the association 137
 - logical columns
 - creating or editing, about 135
 - creating, about 134
 - creating, procedure 135
 - logical column, unmapping from its source 141
 - logical complex join, creating 159
 - Logical Foreign Key dialog box, about using to specify a driving table 162
 - logical foreign key, creating 158
 - logical joins, specifying for driving table (procedure) 162
 - logical object, displaying all that map to physical tables 163
 - Logical Table dialog box
 - foreign key, editing 134
 - key, specifying 133
 - new logical table source, adding 131
 - logical table joins
 - See also* Business Model and Mapping layer, working in
 - about 157
 - Business Model Diagram, displaying 160
 - creating, about 157
 - driving table, about specifying 162
 - driving table, about specifying and query optimization 163
 - driving table, controlling and tuning performance 162
 - driving table, specifying for logical joins (procedure) 162
 - logical complex join, creating 159
 - logical foreign key, creating 158
 - logical object, displaying physical tables that map to 163
 - logical table sources
 - general properties, defining 138
 - removing a table as a source 139
 - Replace Wizard, using to replace tables or columns 235
 - Where clause filter, using to constrain physical tables 250
 - working with, about 137
 - logical table sources, adding and editing 131
 - logical tables
 - columns, about 44
 - creating by dragging and dropping 130
 - creating explicitly 130
 - creating, ways to 129
 - foreign key, editing 134
 - key, specifying 133
 - new logical table source, adding 131
 - working with, about 129
 - LOGLEVEL system session variable, about 330
 - Lower string function, about 449
- ## M
- main window, repository parts
 - described 51
 - Manage menu, described and functions (table) 53
 - many-to-many relationships, about using bridge tables 46
 - MASTER_SERVER parameter, about 346
 - math functions, list of 453
 - Mavg running aggregate function, about 439
 - Max aggregate function, about 434
 - measure
 - dimension, associating with 136

- dimension, removing the
 - association 137
- measure column
 - behavior with aggregate functions 423
 - default aggregation rule, specifying
 - a 136
 - example 424
- measure column, specifying a default aggregation rule 136
- Median aggregate function, about 434
- member hierarchies
 - Query Repository dialog box, using to view 370
 - Security Manager, viewing in 370
- metadata repository, about 32
- metadata, importing
 - about 320
 - connections, about 84
- Min aggregate function, about 435
- Minute calendar date/time function, about 463
- Mod math function, about 457
- modes
 - See offline mode; online mode
- Month calendar date/time function, about 463
- Month_Of_Quarter calendar date/time function, about 464
- MonthName calendar date/time function, about 464
- More tab
 - joins diagrams, using to set default window size 69
 - scrolling speed, using to set 69
- MSUM running aggregate function, about 439
- multi-database joins, about 118
- multidimensional
 - dimensions for a cube from a multidimensional data source 143
- multidimensional data source
 - Ask DBMS, about availability of 91
 - connection pool properties, setting 99

- properties, about setting 99
- multithreaded architecture, about 32

N

- Now calendar date/time function, about 464
- NQClusterConfig.INI file, setting parameters for Cluster Server feature 348
- nQLogViewer utility
 - log records, interpreting 276
 - query log file, using to view 274
- NQQueryStats.log, about 284
- NQSCONFIG.INI file
 - entry, adding an 177
 - queries, setting to collect more information about 283
 - tuning parameters, about 287
- NQServer.log, and NQCluster.log, about opening and examining 350
- NTile aggregate function, about 435
- numeric literals, about 475

O

- Object Type option, using to create virtual physical tables 100
- Octet_Length string function, about 449
- ODBC
 - calls from client applications, list of 321
 - client tools and data sources, about providing connectivity to 320
 - connection, about physical metadata imports 84
 - data source names, configuring 317
 - metadata, importing 321
 - query and reporting tools, about connecting with 321
- offline mode
 - cache invalidation, implications for 310
 - repository, opening in 60
- one-to-many relationships, about primary key-foreign key relationship (diagram) 42

- online help, accessing 60, 75
- online mode
 - cache invalidation, implications for 310
 - changes, making available and saving to disk 64
 - Check In Changes dialog box, about using and tasks 64
 - consistency check, about passing 63
 - consistency, checking repository for 63
 - repository, opening in 61
- Options dialog box, using to set general preferences 65
- Oracle database
 - note, about importing foreign keys from 83
- ORDER BY clause, about 421
- P**
- parallel content descriptions, examples and discussion 256
- password
 - changing 382
 - Siebel Analytics Server Administration account 364
- Percentile aggregate function, about 436
- performance
 - database hints, about resulting in better query performance 125
 - server configuration and tuning 287
- permissions
 - See also* privileges
 - adding or editing 70
 - limiting queries by filtering (procedure) 387
 - limiting queries by maximum run time (procedure) 386
 - limiting queries by number of rows received (procedure) 386
 - limiting queries by objects (procedure) 385
 - limiting queries by time periods (procedure) 386
 - repository objects, about setting for 70
- physical column, creating or editing 105
- physical database model, understanding
 - aggregate navigation, about setting up 43
 - contents of physical database, understanding 43
 - dimensional schemas, about and advantages 38
 - dimensional schemas, about star schema (diagram) 39
 - entity-relationship (E-R schemas), about 37
 - entity-relationship (E-R) schemas, performance for queries for historical analysis 37
 - physical models, types of 37
 - primary-key-foreign key relationships, about and diagram 42
- Physical Diagram
 - command, about using 163
 - displaying 121
 - editor, about using to specify multi-database joins 118
 - foreign key join or complex join, defining 122
 - physical joins, about defining 121
- physical joins
 - See also* Physical layer, working in about 117
 - complex joins, about 120
 - fragmented data, about and example 118
 - Joins Manager, using to define 120
 - multi-database joins, about 118
 - note, about imported key and foreign key joins 117
 - primary key and foreign key relationships, about 119
 - tip, about avoiding unnecessary joins 119
- physical joins, defining in the Physical Diagram 121
- Physical layer

- creating and maintaining, about 81
 - physical layer objects, about 81
 - queries, specifying types of sent to a database 90
- Physical layer, described 76
- Physical layer, working in
 - See also* physical joins
 - catalog, creating or editing 115
 - column mapping, removing 140
 - connection pool, creating or editing 93
 - database hints, about 123
 - database hints, database objects that
 - accept hints (table) 124
 - database object, creating or editing 88
 - database type, restoring default entries for 91
 - Feature table entries, changing using Ask DBMS 91
 - logical columns, mapping to physical columns 139
 - multidimensional data source, setting properties for 99
 - physical joins, defining with the Joins Manager 120
 - query type, locating 91
 - schema folders, working with 115
 - XML data source, setting properties for 97
- physical schemas
 - importing from delimited file (procedure) 86
 - importing from ODBC (procedure) 85
 - importing, about 82
- physical tables
 - creating or editing 103
 - overview 100
 - Physical Table dialog box, completing Columns and Keys tabs 106
 - virtual physical tables, creating using the Object Type option 100
 - XML data source, setting properties for 114
- Pi math function, about 457
- PORTALPATH system session variable, about 330
- Position string function, about 449
- Power math function, about 457
- preferences
 - Cache Manager, changing column order 69
 - Cache Manager, selecting columns to display 69
 - general preferences, setting 65
 - join diagrams, setting default window size 69
 - repository objects, specifying appearance
 - in alphabetical order 68
 - scrolling speed, setting 69
- presentation catalogs
 - Alias tab, using 174
 - caution, about moving columns into presentation catalog folders 170
 - copy business model with presentation catalog utility 236
 - creating (procedure) 169
 - Presentation Catalog dialog box, described 168
 - presentation tables, deleting 170
 - table, reordering in the Presentation layer 170
 - tables, sorting in alphanumeric order 170
 - working in, about 168
- presentation columns
 - Alias tab, using 174
 - creating or editing (procedure) 172
 - Presentation Column dialog box, described (table) 172
 - Repository Documentation utility, using to map to logical and physical columns 241
 - working with, about 172
- Presentation layer
 - about creating 165
 - about creating and maintaining 165

- Presentation Layer dialog box, using the
 - Alias tab 174
 - Presentation layer, creating
 - See also* Presentation layer, working in business models, copying to the Presentation layer 166
 - columns, about removing unneeded or unwanted 166
 - logical keys, about exporting in the Presentation Catalog 167
 - presentation columns, renaming 167
 - Presentation layer repository layer, about 77
 - presentation tables, renaming 166
 - Presentation layer, working in
 - See also* Presentation layer, creating
 - Alias tab, using 174
 - Presentation Catalog dialog box, described 168
 - presentation catalogs, about working in 168
 - presentation catalogs, creating 169
 - Presentation Column dialog box, described (table) 172
 - presentation column, creating or editing 172
 - presentation column, deleting 173
 - presentation column, reordering 173
 - presentation columns, working with 172
 - Presentation layer, about and example 167
 - Presentation Tables dialog box, described (table) 171
 - presentation tables, creating 171
 - presentation tables, deleting 170
 - table, reordering 170
 - tables, sorting in alphanumeric order 170
 - presentation tables
 - Alias tab, using 174
 - column, deleting 173
 - creating (procedure) 171
 - presentation column, reordering 173
 - Presentation Tables dialog box, described (table) 171
 - primary key
 - foreign key, relationship with 119
 - specifying 108
 - primary key-foreign key relationship
 - about and diagram 42
 - PRIMARY_CONTROLLER parameter, about 346
 - privileges
 - query privileges, about controlling and activities 384
 - Project, Manage menu option, described 54
- Q**
- Quarter_Of_Year calendar date/time function, about 465
 - queries
 - aggregate functions, rules for 425
 - database, specifying types sent to 90
 - Leading hint, about using to build the join order 125
 - query caching, enabling to improve performance 288
 - query privileges, controlling and activities 384
 - query type, locating 91
 - query caching
 - about 289
 - advantages of 289
 - cache event processing with an event polling table 302
 - cache hits 298
 - cache hits, information about 298
 - cache management strategy, choosing 295
 - Cache Manager, about and opening 311
 - Cache Manager, options settings (table) 311
 - cache storage, configuring 293
 - cache strategies 298

- cost of caching, about 290
 - disabling for system 295
 - disabling query caching 295
 - enabling query caching 294
 - global cache information,
 - displaying 312
 - improve performance, enabling to 288
 - invalidation after offline repository changes 310
 - note, about query references with
 - different persistence times 105
 - parameters to control query caching,
 - location of 289
 - purge cache options 313
 - purging cache 313
 - refresh interval, setting for XML data
 - sources 315
 - security attribute, about enforcing 290
 - suite of queries, about running 301
 - query environment, administering
 - See also* server configuration and tuning;
 - usage tracking, administering
 - collecting more information, about 283
 - query log, administering 272
 - server, about connecting to 272
 - Siebel Analytics Server, shutting down in
 - UNIX 271
 - Siebel Analytics Server, shutting down in
 - Windows 269
 - Siebel Analytics Server, shutting down
 - using the Administration Tool 271
 - Siebel Analytics Server, starting in
 - UNIX 267
 - Siebel Analytics Server, starting in
 - Windows 265
 - usage tracking, administering 277
 - query execution privileges 384
 - query log, administering
 - about 272
 - file size, controlling 272
 - log viewer utility, interpreting the log records 276
 - log viewer utility, using 274
 - logging levels
 - about enabling 273
 - described (table) 273
 - setting a user's 274
 - logging system, configuring 272
 - user's logging levels
 - disabling 274
 - query repository
 - new object, creating (procedure) 181
 - procedure 181
 - Query Repository dialog box
 - about using 180
 - member hierarchies, using to view 370
 - object, reporting on and viewing the
 - results online 182
 - parent of an object 183
 - searching for object based on name 182
 - Show Qualified Name 182
 - type of object 183
 - Query Repository Filter dialog box
 - about and accessing 184
 - filter, constructing 184
 - filter, constructing to view all database references in a business model 184
 - filter, constructing to view all
 - Presentation layer columns mapped to a logical column 185
 - query specification (SELECT statement),
 - about 418
 - query tool, about connecting with 321
- ## R
- Radians math function, about 458
 - Rand math function, about 458
 - RandFromSeed math function, about 458
 - Rank aggregate function, about 436
 - RCOUNT running aggregate function,
 - about 442
 - refresh interval, setting for XML data
 - sources 315
 - Rename Wizard, about and starting 238

- Rename Wizard, using to rename
 - Presentation layer and Business Model and Mapping layer tables and columns 238
- Repeat string function, about 450
- Replace string function, about 450
- Replace Wizard, about and starting 235
- reporting tool, about connecting with 321
- repositories
 - Administration Tool, about using to create and edit repositories 32
 - comparing repositories 186
 - comparing, turning off Compare Mode 188
 - Execute UDML utility, using to create objects 237
 - LDAP authentication, configuring 371
 - LDAP, importing users and groups into 373
 - making changes to and implications for cache 310
 - merging repositories, about and process 188
 - merging version of Siebel Analytics Repository (procedure) 189
 - new group, adding to 369
 - new user, adding to 362
 - synchronizing 179
 - synchronizing and updating (procedure) 179
- Repository Documentation utility, about and starting 241
- Repository Import Wizard, about and using 179
- repository mode
 - See also* individual repository entries
 - business model within repository, checking consistency of 63
 - changes, making available and saving to disk 64
 - Check In Changes dialog box, about using and tasks 64
- Load all objects option, about
 - selecting 62
- note, editing while repository is being loaded 61
- offline mode, opening repository in 60
- online mode, about passing consistency check 63
- online mode, checking consistency of a repository 63
- online mode, opening repository in 61
- repository objects
 - See also* individual repository entries
 - alphabetical order, specifying in 68
 - note, about naming objects Administrator 79
 - permissions, adding or editing 70
 - permissions, setting 70
- repository variables
 - See also* individual repository entries and initialization blocks
 - about 325
 - cache purging considerations 327
 - Default initializer column, about value 342
 - dynamic repository variables, about 327
 - dynamic repository variables, example 327
 - static repository variables, about 326
 - static repository variables, example 326
 - static repository variables, using in an expression 327
 - static repository variables, using in expression builder 326
 - uses for static repository variables 326
- repository, managing metadata
 - See also* individual repository entries and Query Repository dialog box
 - complex filter, about constructing 185
 - filter, constructing to view all Presentation layer columns mapped to a logical column 185
 - note, about constructing more than one filter 185

- query results, constructing a filter for 184
 - repository, setting up
 - Administration Tools, repository components in 75
 - connection pool, about creating and configuring 91
 - data source, about defining 177
 - new repository, creating 79
 - NQSConfig.INI file, adding entry 177
 - online help, accessing 60, 75
 - physical schemas, about importing 82
 - saving 78, 128, 165
 - saving, checking consistency, and correcting errors 176
 - setup checklist (table) 78
 - Siebel Analytics Server, about starting 178
 - testing and refining, about 178
 - user community, about publishing to 178
 - REPOSITORY_PUBLISHING_DIRECTORY parameter, about 346
 - REQUESTKEY system session variable, about 331
 - Right string function, about 451
 - RMAX running aggregate function, about 443
 - RMIN running aggregate function, about 444
 - rollover, described 283
 - Round math function, about 458
 - row count feature
 - updating for tables and columns 71
 - row counts
 - displaying 71
 - updating 72
 - row counts in physical layer, displaying 71
 - Row-Wise Initialization feature
 - about and example 334
 - variable with a List of Values, initializing 335
 - RSUM running aggregate function, about 441
 - running aggregate functions
 - about 439
 - Mavg, about calculating a moving average 439
 - MSUM, about calculating a moving sum 439
 - RCOUNT, about counting number of input records 442
 - RMAX, about showing the maximum values of a set of records 443
 - RMIN, about showing the minimum values based on a set of records 444
 - RSUM, about calculating a running sum 441
- S**
- sample scripts, locating and example 477
 - schema folders
 - schema object, creating 115
 - schemas
 - See also* schema folders
 - physical schemas, about importing 82
 - physical schemas, importing from delimited file (procedure) 86
 - physical schemas, importing from ODBC (procedure) 85
 - scrolling speed, setting 69
 - Second calendar date/time function, about 465
 - SECONDARY_CONTROLLER parameter, about 346
 - security
 - See also* authentication options; privileges bypassing 383
 - group privileges, about granting and examples 366
 - groups, about creating and example 366
 - groups, about working with 365
 - LDAP authentication, configuring 371
 - LDAP, about importing users and groups 371

- member hierarchies, viewing in the Query Repository dialog box 370
- member hierarchies, viewing in the Security Manager 370
- predefined administrators group 366
- privileges example (diagram) 368
- repository, adding a new group to 369
- repository, adding new user to 362
- repository, importing LDAP users and groups into 373
- Siebel Analytics Server Administration
 - account password 364
- Siebel Analytics Server Administrator
 - account, about 364
- Siebel Analytics Server security,
 - about 75
- user accounts, about working with 362
- user privileges and group privileges
 - example (diagram) 367
- Security Manager
 - See also* security
 - member hierarchies, using to view 370
- Security, Manage menu option,
 - described 53
- SELECT list syntax
 - about and syntax 420
 - FROM clause syntax, about 420
 - GROUP BY clause syntax, about 421
 - ORDER BY clause syntax, about 421
 - WHERE clause syntax, about 420
- SELECT statement
 - about and basic syntax 418
 - conditional expressions, list of 427
 - queries and aggregate functions, rules for 421
 - query specification 418
 - Select list syntax 420
 - SQL logical operators 427
 - usage notes 419
 - WHERE clause 420
- server configuration and tuning
 - See also* query environment, administering
- aggregate tables, about using 287
- databases, importance of configuring, tuning, and indexing 288
- NQSCONFIG.INI parameters. about using for tuning 287
- query caching, about enabling to improve performance 288
- server session management
 - See* server configuration and tuning; Session Manager
- Session Manager
 - See also* query environment, administering
 - active query, killing 286
 - disconnecting a user from a session 286
 - Request Window fields (table) 285
 - Session Window fields (table) 285
 - session, viewing 286
 - update speed, controlling 284
 - using, about 284
- session variables
 - See also* initialization blocks
 - about 325
 - creating or editing (procedure) 336
 - initializing, about 334
 - nonsystem session variables, about using 331
 - row-wise initialization, about and example 334
 - system session variables, about using 329
 - system session variables, table of 330
 - using for authenticating users 329
- Sessions, Manage menu option,
 - described 53
- shutting down Siebel Analytics Server Administration Tool, using to shut down server 271
- UNIX, shutting down server 271
- Windows, shutting down from a command prompt 270
- Windows, shutting down from the Services applet 269

- Siebel Analytics Client testing tool 33
- Siebel Analytics Event Tables utility, about and starting 236
- Siebel Analytics Scheduler, about interface to 53
- Siebel Analytics Server
 - See also* Cluster Server feature
 - Administration Tool, shutting down using 271
 - automatic startup, configuring for 266
 - business model, about interface to physical databases 37
 - connecting to, about 272
 - fails to start 268
 - internal authentication, about 382
 - nonlocal files, about accessing 391
 - note, about changing repository in online mode and attempting to stop server 65
 - password, changing 382
 - repository variable initialization blocks, re-initializing 342
 - UNIX, running the shutdown script 271
 - UNIX, running the Siebel startup script 267
 - user ID, changing 267
 - user IDs and passwords, about and storage of 382
 - Windows, configuring for automatic startup 266
 - Windows, shutting down from a Windows command prompt 270
 - Windows, shutting down from the Services applet 269
 - Windows, starting from the Services applet 265
- Siebel Analytics Server Administrator account, about and password 364
- group, about 366
- Siebel Analytics Server security, about 75
- Siebel Analytics Server XML Gateway
 - See* XML Gateway
- Sign math function, about 459
- Sin math function, about 459
- single table model, about and creating 49
- SKIN system session variable, about 331
- snowflake models, about dimensional hierarchies 46
- Sort Objects tab, using to specify repository objects in alphabetical order 68
- SQL 92 functions
 - datetime literals, about 474
 - NTILE function, about 436
- SQL FROM clause syntax, about 420
- SQL functions
 - aggregate functions, about 430
 - calendar date/time functions, about 460
 - conversion functions, about 471
 - datetime literals, about 474
 - expression literals, about 474
 - NTILE function, about 436
 - running aggregate functions 439
 - string functions, about 445
 - system functions, about 474
- SQL logical operators, list of 427
- SQL queries, about selecting the Export logical keys option 167
- SQL statement
 - database hints, about 123
 - database hints, creating 125
 - database objects that accept hints (table) 124
- SQL syntax and semantics
 - conditional expressions, list of 427
 - queries and aggregate functions, rules for 421
 - Select list syntax 420
 - Select statement, about and basic syntax 418
 - Select usage notes 419
 - SQL logical operators 427
- SQL WHERE clause syntax, about 420
- Sqrt math function, about 459
- star schema
 - about and diagram 39
 - dimensional hierarchies, about 46

- static repository variables
 - See also* initialization blocks
 - about 326
 - example 326
 - expression builders, using in 326
 - expression, using in 327
- status bar, showing 67
- StdDev aggregate function, about 437
- STORAGE_DIRECTORY parameter
 - user tracking log files, selecting an output location for 279
- string functions
 - about 445
 - Abs, about calculating the absolute value 453
 - Acos, calculates the arc cosine of a numerical expression 453
 - ASCII, about converter single character string to 445
 - Asin, about calculating the arc sine of a numerical expression 453
 - Atan, about calculating the arc tangent of a numerical expression 454
 - Atan2, about calculating the arc tangent of y/x 454
 - Bit_Length, about returning length in bits 445
 - Ceiling, about rounding a noninteger numerical expression 454
 - Char, about converting a numerical value 445
 - Char_Length, about returning length in number of characters 446
 - Concat, about forms of function 446
 - Cos, about calculating the cosine of a numerical expression 455
 - Cot, about calculating the cotangent of a numerical expression 455
 - Degrees, about converting an expression from radians to degrees 455
 - Exp, about sending the value e to the power specified 456
 - Floor, about rounding a noninteger numerical expression 456
 - Insert, about inserting a character string 447
 - Left, about returning characters from the left of a string 447
 - Length, about returning the length in number of characters 448
 - Locate, about returning the numerical position of the character_expression1 448
 - LocateN, about returning the numerical position of the character_expression1 448
 - Log, calculated the natural logarithm of an expression 456
 - Log10, about calculating the base 10 logarithm of an expression 456
 - Lower, about converting a character string to lower case 449
 - Mod, about dividing the first numerical expression 457
 - Octet_Length, about returning the bits in base 8 units 449
 - Pi, about returning the value of pi 457
 - Position, about returning the numerical position of the character_expression1 449
 - Power, about taking the first numerical expression to the power specified 457
 - Radians, about converting from degrees to radians 458
 - Rand, about returning a pseudo-random number 458
 - RandFromSeed, about returning a pseudo-random number from a seed value 458
 - Repeat, returns a specified expression n times 450
 - Replace, about replacing specified characters 450

- Right, about returning a specified number of characters from the right of the string 451
 - Round, about rounding a numerical expression to n digits 458
 - Sign, about returning a value of 1, -1, or 0 459
 - Sin, calculated the sine of a numerical expression 459
 - Sqrt, about calculating the square root of the numerical expression argument 459
 - Substring, about creating a new string starting from a fixed number 451
 - Tan, about calculates the tangent of a numerical expression 460
 - TrimBoth, about stripping specified leading and trailing characters 451
 - TrimLeading, about stripping specified leading characters 452
 - TrimTrailing, about stripping specified trailing characters 452
 - Truncate, about truncating a decimal number 460
 - Upper, about converting a character string to uppercase 452
 - Substring string function, about 451
 - Sum aggregate function, about 437
 - SumDistinct aggregate function, about 438
 - symbols, described (table) 56
 - Synchronize Aliases utility, about and procedure 234
 - system
 - session variables, about and LDAP authentication 376
 - SQL functions, about 474
 - variables, about and external table authentication 378
- T**
- tables
 - content definitions, creating 141
 - event polling table, identifying 236
 - one-to-many relationship, about and diagram 42
 - Tan math function, about 460
 - TCP/IP, client/server communication method changed 319
 - text strings, using the Externalize Strings utility to translate 237
 - Times Series Wizard, about and starting 234
 - TimestampAdd calendar date/time function, about 465
 - TimeStampDiff calendar date/time function, about 468
 - toolbar
 - docking 54
 - on/off, toggling 54
 - Tools menu, described 54
 - TopN aggregate function, about 438
 - TrimBoth string function, about 451
 - TrimLeading string function, about 452
 - TrimTrailing string function, about 452
 - troubleshooting
 - event polling table 309
 - Siebel Analytics Server fails to start 268
 - Truncate math function, about 460
 - Turn Off Compare Mode, enabling 188
- U**
- UDML utility, starting 237
 - Unified Logon, about support of 375
 - Universal Data Modeling Language, using the Execute UDML utility to create objects 237
 - UNIX
 - shutting down the Siebel Analytics Server 271
 - starting the Siebel Analytics Server 267
 - Update Physical Layer wizard, about and starting 239
 - Upper string function, about 452
 - usage tracking log files

- sample scripts, locating and
 - example 477
 - usage tracking data (table) 478
 - usage tracking, administering
 - See also* Session Manager; usage tracking
 - log files
 - about and example 277
 - file naming conventions, about and example 280
 - note, about error accessing usage tracking output file 280
 - output file, about and schema described (table) 281
 - output file, column behavior in 282
 - output file, format of 281
 - output location, selecting 279
 - performance considerations, about and checkpoints described 283
 - queries, collecting more information about 283
 - user community, about publishing to 178
 - user ID, changing for the Siebel Analytics Server 267
 - user interface components
 - Edit menu, described 53
 - File menu, described 53
 - Help menu, described 54
 - icons and symbols (table) 56
 - keyboard shortcuts (table) 55
 - main window, described 51
 - Manage menu, described 53
 - toolbar functions, described 54
 - Tools menu, described 54
 - View menu, described 53
 - Window menu, described 54
 - USER session system variable, defining for LDAP authentication 377
 - User system function, about 474
 - USER system session variable, about 330
 - USER_LOG_FILE_SIZE parameter, using to control query log file size 272
 - users
 - See also* privileges
 - LDAP, using to import users 371
 - new user, adding to repository 362
 - Siebel Analytics Administration account, about and password 364
 - user accounts, about 362
 - utilities
 - copy business model with presentation catalog utility, about and procedure 236
 - Execute UDML utility, about and starting 237
 - Externalize Strings utility, about and starting 237
 - log viewer utility, interpreting the log records 276
 - log viewer utility, using 274
 - Repository Documentation utility, about and starting 241
 - Siebel Analytics Event Tables utility, about and starting 236
 - Synchronize Aliases, about and procedure 234
- ## V
- VALUEOF() conversion function, about 472
 - ValueOf() expressions, using 407
 - Variables, Manage menu option, described 54
 - variables, using
 - See also* Initialization Block dialog box; initialization blocks
 - Default initializer column, about value 342
 - dynamic repository variables, about and example 327
 - new variables, creating 332
 - nonsystem session variables, about using 331
 - session variables, about 328
 - Siebel Analytics Server, re-initializing when it starts 342

- static repository variables, about and example 326
 - static repository variables, using variables in expression builders 326
 - system session variables, about and LDAP authentication 376
 - system session variables, about using 329
 - system session variables, table of 330
 - system variables, about and external table authentication 378
 - Variable Manager, about and classes of variable 325
 - View menu, described 53
 - virtual physical tables, creating using the Object Type option 100
- W**
- WEBGROUPS system session variable, about 330
 - Week_Of_Quarter calendar date/time function, about 470
 - Week_Of_Year calendar date/time function, about 470
 - WHERE clause
 - filter, about and example 250
 - syntax, about 420
 - Window menu, described 54
 - Windows
 - Cluster Controller and Analytics Server, starting manually 350
 - NT and 2000 Administrator account, about and Siebel Analytics Server Administrator account 364
 - NT and 2000, about support of Unified Logon 375
 - NT and 2000, user ID to access nonlocal files 391
 - ODBC data source names, configuring 317
 - shutting down Siebel Analytics Server from a command prompt 270
 - shutting down Siebel Analytics Server from the Services applet 269
 - Siebel Analytics Server, configuring for automatic startup 266
 - starting Siebel Analytics Server from the Services applet 265
 - wizards
 - Calculation wizard, about 66
 - Rename Wizard, about and starting 238
 - Replace Wizard, about and starting 235
 - Repository Import Wizard, about and using 179
 - Time Series Wizard, about and starting 234
 - Update Physical Layer Wizard, about and starting 239
 - workspace, Administration Tool
 - Presentation Catalog dialog box, about using Presentation Table tab 168
 - Presentation Tables dialog box, about using Columns tab 171
- X**
- XML data source
 - Ask DBMS, about availability of 91
 - connection pool properties, setting 97
 - physical table, setting properties for 114
 - properties, about setting 97
 - query output format settings, specifying 98
 - refresh interval, setting 315
 - XML Gateway
 - See also* Data Mining Adapter; XML ODBC database type; XML Gateway
 - about using 391
 - example 394
 - example, more complex 396
 - HTML tables, accessing 402
 - HTML tables, example 402
 - XML data, importing using the XML Gateway 392
 - XML examples 411
 - XML ODBC database type

- See also* Data Mining Adapter; XML ODBC database type; XML Gateway
 - example 410
 - importing XML data 409
 - XML data sources, about accessing 409
 - XML examples 411
 - XML, using as a data source
 - See also* Data Mining Adapter; XML ODBC database type
 - about 389
 - example, more complex 396
 - HTML tables, accessing 402
 - HTML tables, example 402
 - importing data using the XML Gateway 392
 - XML examples 411
 - XML Gateway example 394
 - XML Gateway, about using 391
 - XML URL, locating 389
 - XPath expressions, support of 390
 - XSL transformation files (XSLT), support of 390
 - XMLA
 - Siebel Analytics, use of 87
- Y**
- Year calendar date/time function, about 471