SIEBEL ANALYTICS SERVER ADMINISTRATION GUIDE

VERSION 7.5.3

JULY 2003

12-FRKILT

Contents

Introduction

How This Guide Is Organized 14

Revision History 15

Chapter 1. Overview of the Siebel Analytics Server

The Siebel Analytics Server and Business Modeling 22

Siebel Analytics Server Features 24

Siebel Analytics Server in the Decision Support Environment 26

Siebel Analytics Server and the Data Warehouse 29

Siebel Analytics Server Components 32

Chapter 2. Data Modeling

Understanding the Business Model 38

Understanding the Physical Database Model 40

- Primary Key-Foreign Key Relationships 45
- Knowing the Contents of the Physical Database 46
- Knowing the Aggregate Table Definitions 47

Logical Business Models 48

Understanding Dimensional Models 49

Chapter 3. Administration Tool Basics

User Interface Components 55

Online and Offline Repository Modes 64

Checking the Consistency of a Repository or a Business Model 66

Checking In Changes 67

Setting Preferences 69

Using the Options Dialog Box—General Tab 69

Using the Options Dialog Box—Sort Objects Tab 71

Using the Options Dialog Box—Cache Manager Tab 72

Using the Options Dialog Box—More Tab 72

Setting Permissions for Repository Objects 73

Displaying and Updating Row Counts for Tables and Columns 75

Using the Browse Dialog Box 76

Chapter 4. Setting Up and Working with a Repository

Components of a Repository in the Administration Tool 79

Physical Layer 79

Business Model and Mapping Layer 80

Presentation Layer 81

Security 81

Context-Sensitive Menus 81

The Steps to Setting Up a Repository 82

Creating a New Repository 83

Importing the Physical Schemas 83

Create and Configure the Connection Pool 85

Define Join Conditions on the Physical Schema 85

Create the Business Model 86

Creating the Presentation Layer 86

Save Repository, Check Consistency, and Correct Any Errors 88

Add an Entry in the NQSConfig.INI File 90

Create the Data Source 91
Start the Siebel Analytics Server 91
Test and Refine the Repository 91
Publish to User Community 91
Synchronizing Repositories 92
Managing Repository Metadata 95
Constructing a Filter for Query Results 98
Comparing Repositories 100
Merging Repositories 103

Chapter 5. Working in a Repository’s Presentation Layer

Presentation Layer Objects 109
Working with Presentation Catalogs 109
Working with Presentation Tables 112
Working with Presentation Columns 113
Using the Alias Tab of Presentation Layer Dialog Boxes 115

Chapter 6. Working in a Repository’s Physical Layer

Physical Layer Objects 117
Working with Databases 118
Working with Connection Pools 121
Working with Catalog Folders 128
Working with Schema Folders 130
Working with Physical Tables 131
Using the Physical Table Dialog Box 133
Specifying a Primary Key 136
Working with Physical Columns 136

About Physical Joins 138
Defining Physical Joins with the Joins Manager 141
Defining Physical Joins in the Physical Table Diagram 143
Using Database Hints 145

Chapter 7. Working in a Repository's Business Model and Mapping Layer

Business Model and Mapping Layer Objects 149
Working with Business Models 149
Working with Logical Tables 150
Working with Logical Columns 155
Working with Logical Table Sources (Mappings) 160
Working with Dimensions 165
Creating a Dimension 166
Working with Levels 167
About Logical Joins 175
Defining Logical Joins with the Joins Manager 176
Defining Logical Joins with the Business Model Diagram 178
Specifying a Driving Table 180
Showing Physical Tables That Are Sources of Logical Objects 181

Chapter 8. Administration Tool Utilities and Expression Builder

Utilities and Wizards 183
Time Series Wizard 184
Synchronize Aliases 184
Replace Wizard 185
Copy Business Model with Presentation Catalog 185
Siebel Analytics Event Tables 186
Execute UDML 186
Query Statistics 186

Externalize Strings 187
Rename Wizard 187
Update Physical Layer Wizard 188
Repository Documentation 191
Expression Builder 192

Chapter 9. Setting Up Aggregate Navigation

About Aggregate Navigation 199
Specify the Aggregate Levels for Each Source 200
Create Dimension Sources for Each Level of Aggregated Fact Data 202
Specify Fragmentation Content 204
Dimension-Specific Aggregation Rules 210
Aggregate Table Fragments 211

Chapter 10. Administering the Query Environment

Starting the Siebel Analytics Server 217
Starting the Server from the Services Applet in Windows 217
Configuring the Server for Automatic Startup in Windows 218
Running the Siebel Startup Script in UNIX 219
Changing the User ID in Which the Siebel Analytics Server Runs 219
If the Server Fails to Start 220
Shutting Down the Server 221
Shutting Down the Server from the Services Applet in Windows 221
Shutting Down the Server from a Command Prompt in Windows 222
Running the Siebel Analytics Server Shutdown Script in UNIX 223
Shutting Down the Server Using the Administration Tool 223
Getting Users to Connect to the Server 224
Administering the Query Log 225

Administering Usage Tracking 230
Selecting an Output Location 230
Performance Considerations 234
Collecting More Detailed Information About Queries 235
Server Session Management 236
Server Configuration and Tuning 239

Chapter 11. Query Caching

About the Query Cache 241
Query Cache Architecture 245
Configuring Query Caching 246
Monitoring and Managing the Cache 249
Strategies for Using the Cache 251
Cache Event Processing with an Event Polling Table 255
Setting Up Event Polling Tables on the Physical Databases 256
Making the Event Polling Table Active 260
Populating the Siebel Analytics Server Event Polling Table 262
Troubleshooting Problems with an Event Polling Table 262
Making Changes to a Repository 263
Using the Cache Manager 264
Displaying Global Cache Information 265
Purging Cache 266
About the Refresh Interval for XML Data Sources 268

Chapter 12. Connectivity and Third-Party Tools

Configuring Siebel Analytics ODBC Data Source Names (DSNs) 269
Third-Party Tools and Relational Datasource Adapters 273
Importing Metadata 274
ODBC Conformance Level 275

Chapter 13. Using Variables in a Repository

Using the Variable Manager 277

Using Repository Variables 278

About Session Variables 281

Using System Session Variables 281

Using Nonsystem Session Variables 283

Creating New Variables 284

About Initialization Blocks 286

Initializing Dynamic Repository Variables 286

Initializing Session Variables 288

Creating and Editing Initialization Blocks 290

Tasks Using the Initialization Block Dialog Box—Variable Tab 294

Default Initializer 295

Initialization When the Siebel Analytics Server Starts 295

Execution Precedence 296

Chapter 14. Clustering Siebel Analytics Servers

About the Cluster Server Feature 297

Components of the Cluster Server Feature 298

Implementing the Cluster Server Feature 300

Chronology of a Cluster Operation 304

Using the Cluster Manager 306

Viewing Cluster Information 307

Managing Clustered Servers 312

Performance Considerations 314

Chapter 15. Security

Security Manager 315
Working with Users 316
Working with Groups 319
Importing Users and Groups from LDAP 325
Authentication Options 329
Operating System Authentication 329
LDAP Authentication 330
External Table Authentication 332
Database Authentication 335
Siebel Analytics Server Internal Authentication 336
Order of Authentication 337
Bypassing Siebel Analytics Server Security 337
Query Execution Privileges 339

Chapter 16. Using XML as a Data Source

Overview 343
Locating the XML URL 344
Using the Siebel Analytics Server XML Gateway 346
Siebel Analytics Server XML Gateway Example 348
Accessing HTML Tables 356
Using the Data Mining Adapter 358
Using XML ODBC 364
XML ODBC Example 365
XML Examples 367
83.xml 368
8_sch.xml 369
84.xml 370
Island2.htm 371

Chapter 17. SQL Reference

SQL Syntax and Semantics 373
SELECT Query Specification Syntax 374
SELECT Usage Notes 375
SELECT List Syntax 376
Rules for Queries with Aggregate Functions 377
SQL Logical Operators 383
Conditional Expressions 383
SQL Reference 386
Aggregate Functions 386
Running Aggregate Functions 396
String Functions 402
Math Functions 411
Calendar Date/Time Functions 418
Conversion Functions 429
System Functions 431
Expressing Literals 431

Appendix A. Sample Scripts and Load Procedures for Usage Tracking Data

Locating the Sample Scripts for Usage Tracking Data 436
Extracting to Oracle 437
Extracting to DB2 439
Extracting to SQL Server 441
Description of the Usage Tracking Data 443

Index

Introduction

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.

How This Guide Is Organized

This document is organized into chapters and appendices, each devoted to a particular aspect of the Siebel Analytics Server environment. Siebel administrators should read the chapters and appendices that will apply to their implementation.

- The first two chapters are introductory in nature. They describe how the Siebel Analytics Server, the core component of Siebel Analytics, fits into decision support, Web-based, and client/server decision-making applications, explain some of the benefits of the product, describe the main components of the product, and discuss data modeling concepts that are necessary to create business models for use with the Analytics server.
- Subsequent chapters describe the process of creating and modifying a Siebel Analytics metadata repository, and how to use the Siebel Analytics Server Administration Tool to handle administrative issues and procedures for the Analytics server.
- Other chapters explain how to set up and maintain the query caching subsystem, how to set up a DSN to connect to the Analytics server from other applications and third party tools, how to use variables in a repository, how to set up XML data sources, and security considerations.

Revision History

Siebel Analytics Server Administration Guide

Version 7.5.3

Table 1. Changes Made in Version 7.5.3

Topic	Revision
"Web Clients and Other ODBC Clients" on page 35	Added a note stating that the Siebel Analytics Client testing tool does not support double byte character sets.
"Checking the Consistency of a Repository or a Business Model" on page 66	Clarified differences between error, warning, and informational consistency check messages.
"Importing the Physical Schemas" on page 83	Moved note about not importing foreign keys from an Oracle database into Step 4 of the procedure for importing schemas.
"Merging Repositories" on page 103	Corrected Step 5 in the procedure for merging versions of Siebel Analytics repositories to state that the correct option to select is File > Merge.
"Creating or Editing a Connection Pool" on page 121	Added a definition of the Isolation level field on the General tab of the Connection Pool dialog box. Updated the procedure to select an isolation level if applicable.
"Update Physical Layer Wizard" on page 188	Added procedure for using the Update Physical Layer wizard.
"Output File Format" on page 231	Updated this section to state that the output file format for usage tracking log files is text format.
"Enabling Query Caching" on page 248	Added procedure for enabling query caching.
"Disabling Query Caching" on page 248	Added procedure for disabling query caching.
"Cache Hits" on page 251	Clarified the description of a cache hit.

Table 1. Changes Made in Version 7.5.3

Topic	Revision
"Making Changes to a Repository" on page 263	Updated this topic to include switching between repositories.
Appendix A, "Sample Scripts and Load Procedures for Usage Tracking Data"	Changed the name of the usage tracking table referenced in the sample scripts from NQSACCT to S_NQ_ACCT.
"Locating the Usage Tracking Log Files" and "Understanding the File Format" in Appendix A, "Sample Scripts and Load Procedures for Usage Tracking Data"	Deleted these sections. The information they contained is located in the chapter on administering the query environment.
"Importing into Microsoft Access" in Appendix A, "Sample Scripts and Load Procedures for Usage Tracking Data"	Deleted this section because it is obsolete.
"Appendix B, Accessing International Data"	Deleted this appendix. The information it contained was moved to the international considerations section in *Siebel Analytics Installation and Configuration Guide*.

Additional Changes

- Removed all references to supported databases and replaced with reference to *System Requirements and Supported Platforms* on Siebel SupportWeb.

May 2003 Bookshelf

Table 2. Changes Made in Rev. C for May 2003 Bookshelf

Topic	Revision
"Physical Layer Objects" on page 117	Added note.

January 2003 Bookshelf

Table 3. Changes Made in Rev. B for January 2003 Bookshelf

Topic	Revision
"Displaying and Updating Row Counts for Tables and Columns" on page 75	Moved procedure to Administration Tool Basics chapter.
"Synchronizing Repositories" on page 92	Moved procedure to Setting Up and Working with a Repository chapter.
"Expression Builder" on page 192	Moved topic to Administration Tool Utilities and Expression Builder chapter.
"Defining Logical Joins with the Joins Manager" on page 176	Individuated procedures for creating complex joins and foreign keys.
"Defining Content of Sources" on page 163	Corrected procedure.
"Understanding the Business Model" on page 38	Quality enhancements.
"Understanding the Physical Database Model" on page 40	Quality enhancements.
"Snowflake Schema" on page 44	Corrected Figure 4.
"Using the Options Dialog Box—Sort Objects Tab" on page 71	Added procedure.
"Using the Options Dialog Box—More Tab" on page 72	Added procedure.
"Setting Permissions for Repository Objects" on page 73	Moved topic to Administration Tool Basics chapter.
"Using the Browse Dialog Box" on page 76	Added procedure.

Table 3. Changes Made in Rev. B for January 2003 Bookshelf

Topic	Revision
"Managing Repository Metadata" on page 95	Moved to Setting Up and Working with a Repository chapter.
"Constructing a Filter for Query Results" on page 98	Moved to Setting Up and Working with a Repository chapter.
"Object Types" on page 131	Quality enhancements.
"About Physical Joins" on page 138	Quality enhancements.
"Using the Level Dialog Box—Preferred Drill Path Tab" on page 173	Quality enhancements.
"About Logical Joins" on page 175	Quality enhancements.
"Presentation Column Dialog Box" on page 114	Corrected procedure.
"Using the Alias Tab of Presentation Layer Dialog Boxes" on page 115	Added procedure.
"Administering the Query Environment" on page 217	Updated script names.
"Tasks Using the Initialization Block Dialog Box—Variable Tab" on page 294	Added topics.
"Security Manager" on page 315	Added information about encryption standards.
"SQL Reference" on page 386	Quality enhancements.

November 2002 Bookshelf

Table 4. Changes Made in Rev. A for November 2002 Bookshelf

Topic	Revision
"Working in a Repository's Physical Layer" on page 117	Reorganized content and reworded headings for easier accessibility.

Table 4. Changes Made in Rev. A for November 2002 Bookshelf

Topic	Revision
"Working in a Repository's Business Model and Mapping Layer" on page 149	Reorganized content and reworded headings for easier accessibility.
"Working in a Repository's Presentation Layer" on page 109	Reorganized content and reworded headings for easier accessibility.

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 organization's supply chain. 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.

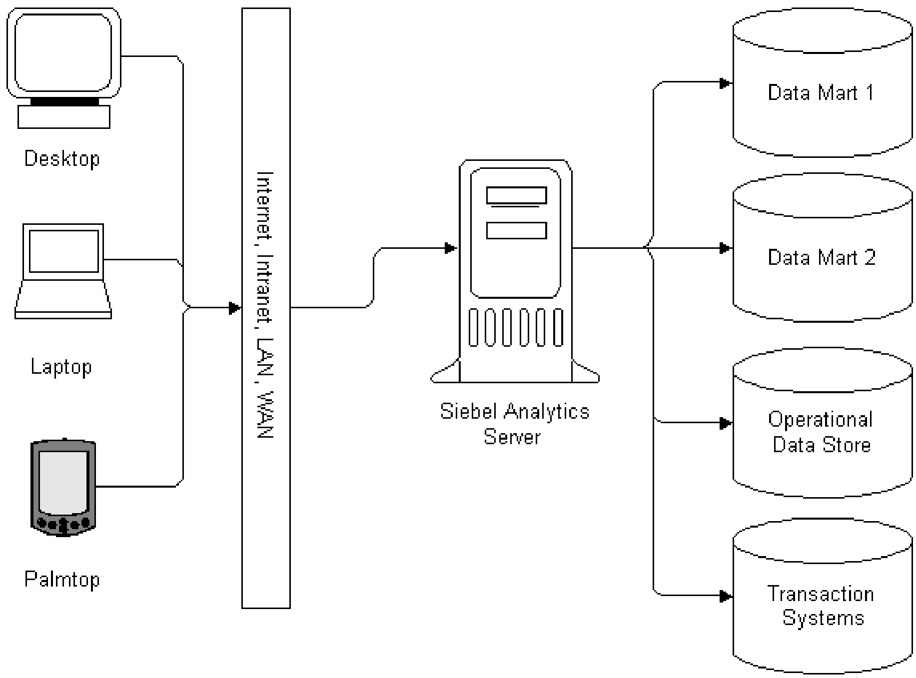


Figure 1. High-Level Environment Diagram

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.

- 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 the users' 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 33 shows the architectural components of the Siebel Analytics Server and how they fit into the enterprise.

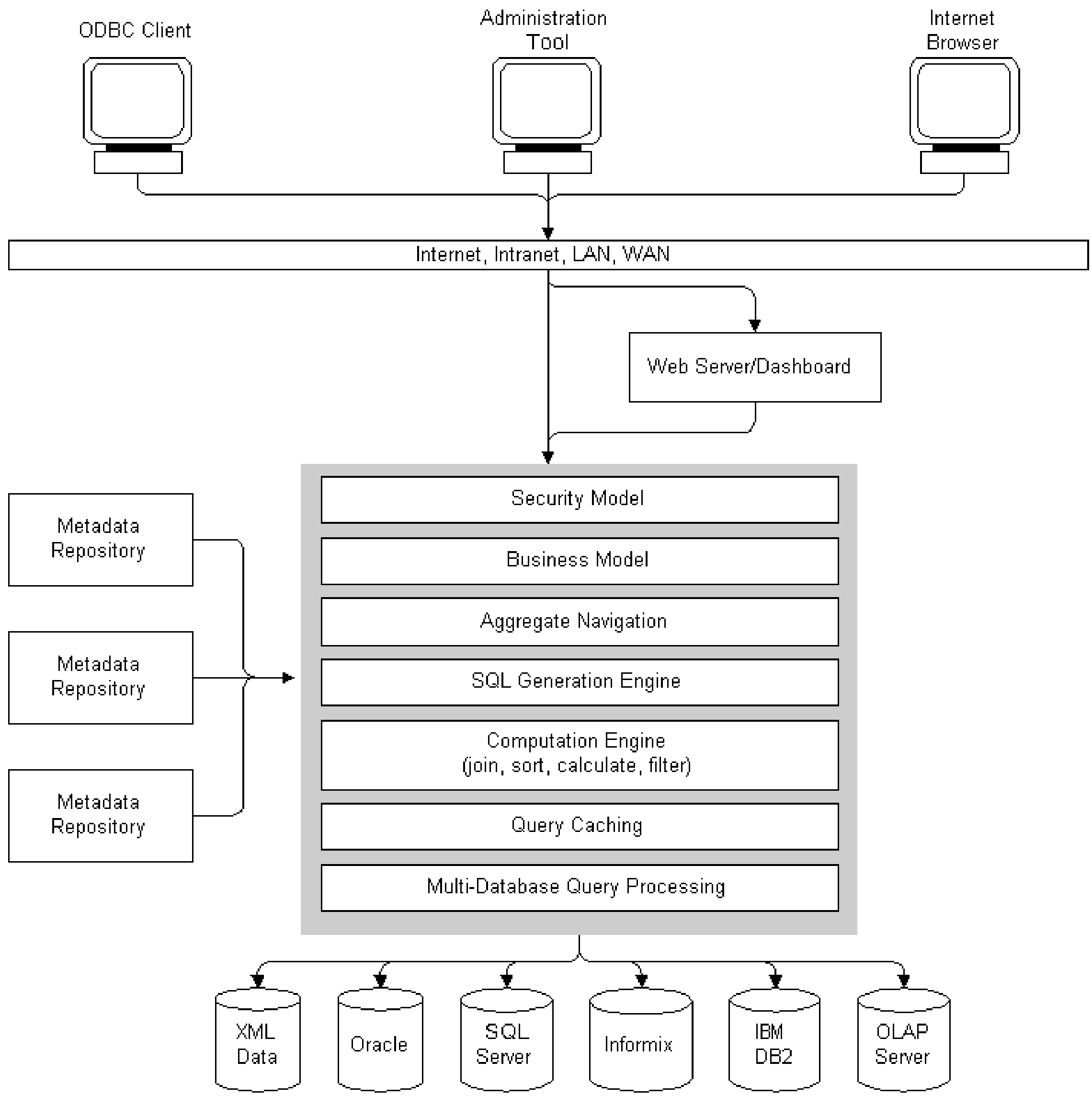


Figure 2. Architectural Components

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.

Administration Tool

The Siebel Analytics Server 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*.

---

2 Data Modeling

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 Siebel Analytics Server 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 Siebel Analytics Server 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.

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.

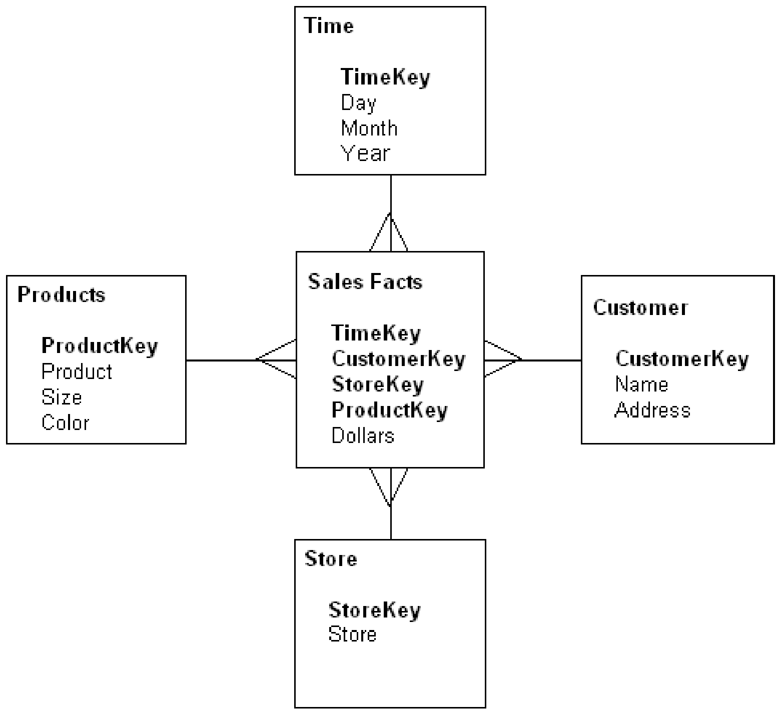


Figure 3. Sample Star Schema

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 47.

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.

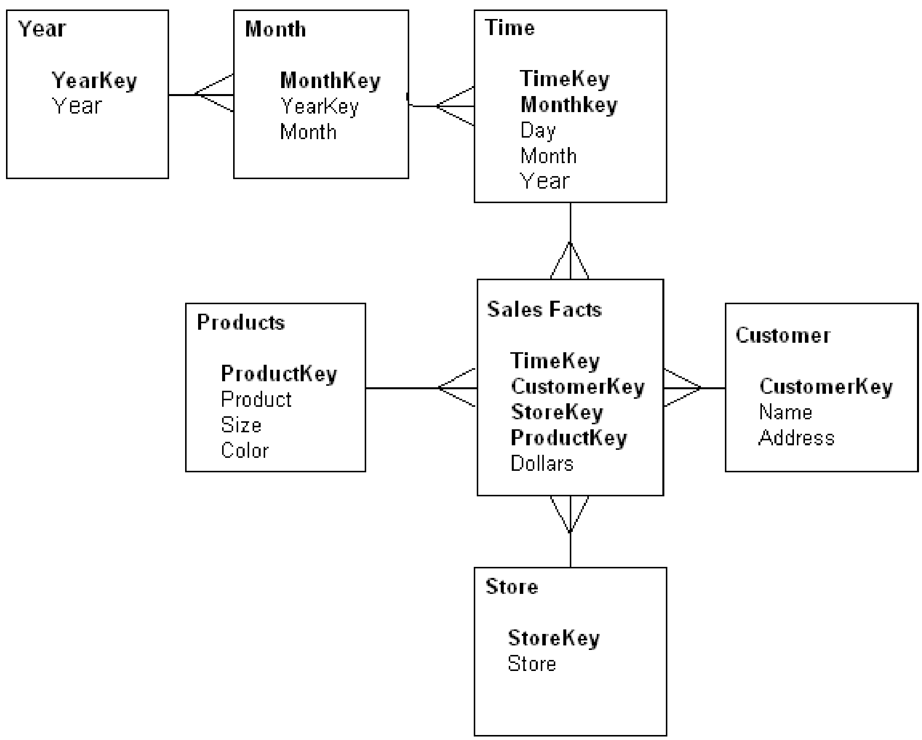


Figure 4. Sample Snowflake Schema

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.

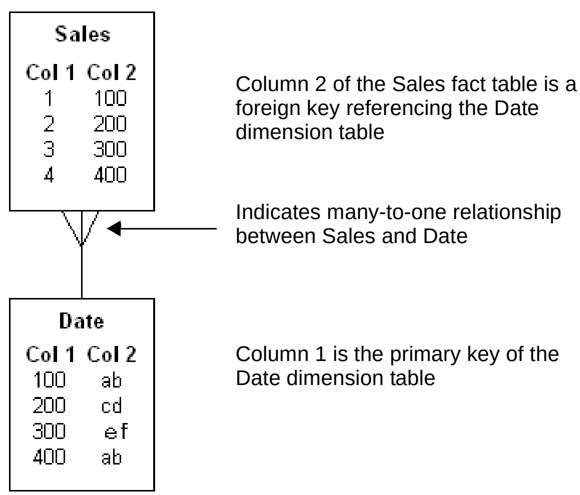


Figure 5. Primary Key-Foreign Key Sample

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 Siebel Analytics Server 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 199.

Logical Business Models

Using the Siebel Analytics Server 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.

Hierarchies are logical levels, usually within a given dimension, that have a one-to-many correspondence between one attribute and another. Hierarchies can get complex and can be transitive; that is, sets of hierarchies can imply other hierarchies. 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 are illustrated in Figure 7 and make up this time dimension.

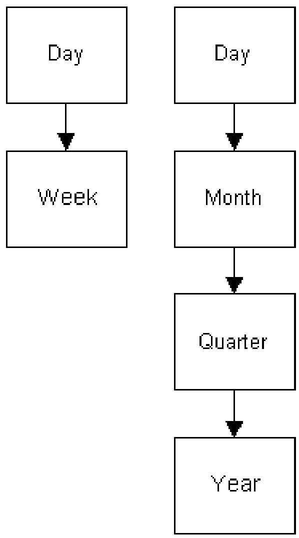


Figure 7. Sample Hierarchy Rollups

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 are excellent for modeling a particular part of a business where there are one-to-many relationships between the dimension tables and the fact tables. Sometimes, however, 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 8.

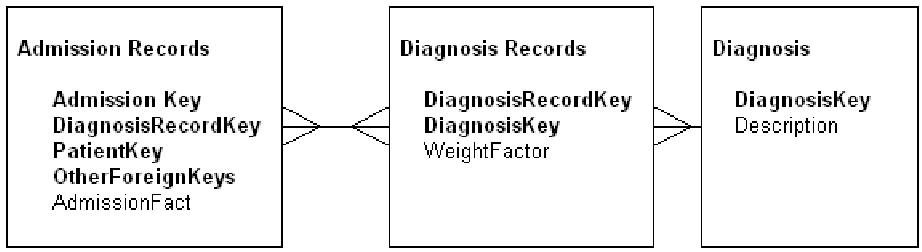


Figure 8. Sample Health Care Schema

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 9.

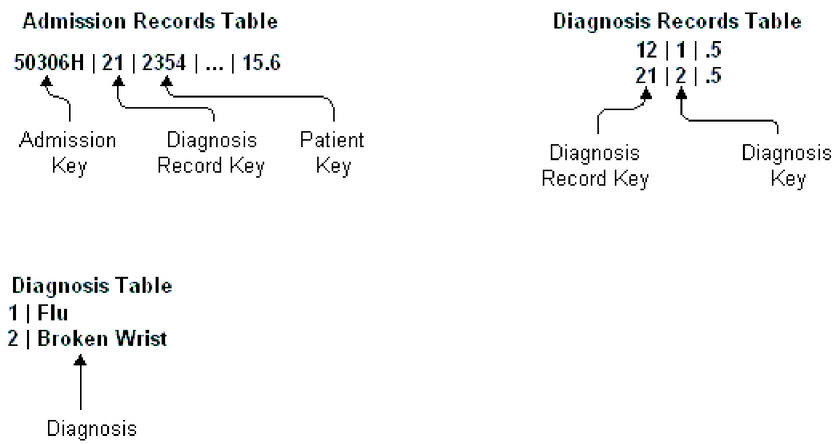


Figure 9. Multiple Diagnoses for One Patient

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 Siebel Analytics Server 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 "Working in a Repository's Presentation Layer" on page 109.

Administration Tool Basics 3

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

User Interface Components

This section includes a description of the following topics:

- "Main Window"
- "Menus" on page 56
- "Toolbar" on page 58
- "Keyboard Shortcuts" on page 59
- "Icons and Symbols" on page 59

Main Window

The main window of the Administration Tool provides a graphical representation of the three parts of a repository, each of which is described briefly in this section.

- **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 10 shows the three layers of a repository, as described in the preceding discussion.

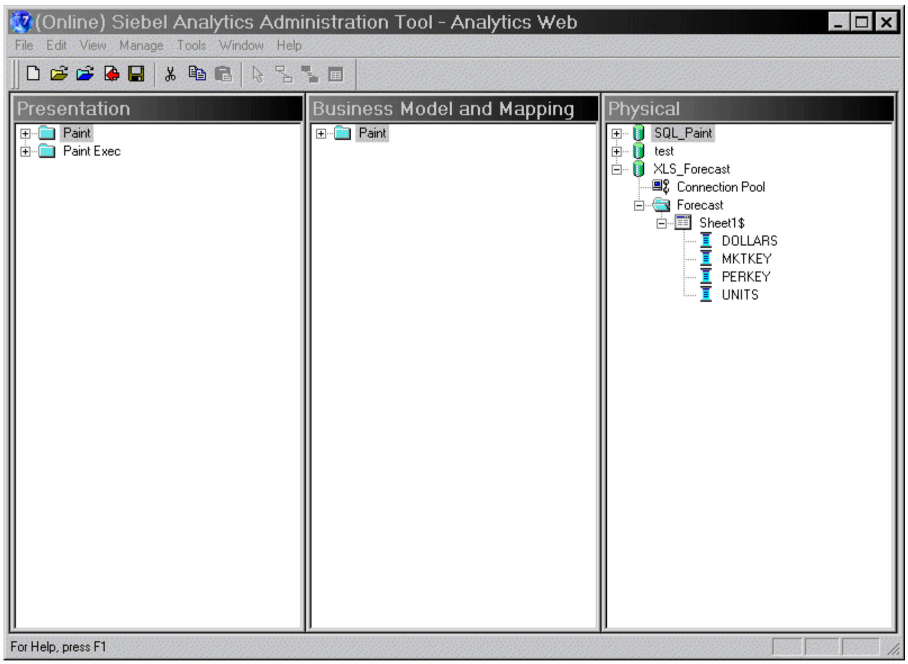


Figure 10. Example Administration Tool Main Window

Menus

The Administration Tool includes the following menus:

File

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

Edit

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

View

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

Manage

The Manage menu allows you to access the management functions described in Table 5.

Table 5. Manage Menu Functions

Menu Option	Description
Job Manager	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.
Session Manager	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.
Cache Manager	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 the cache.
Cluster Manager	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 Manager	This option opens the Security Manager, which displays security information for a repository and provides a central location for security configuration and management.
Joins Manager	This option opens the Joins Manager, where you can work with physical and logical joins.
Variables Manager	This option opens the Variables Manager window, which you can use to create, edit or delete variables.

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 access the online help system for the Administration Tool.

Toolbar

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

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

Table 6. Keyboard Shortcuts

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

Icons and Symbols

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

Table 7. Icons and Symbols

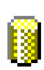
Icon or Symbol	What It Represents
	A database in the physical layer and in the Logons tab of the User dialog box. Identifies a database for which you can specify a user ID and a password for the Siebel Analytics Server to use.
	A database that does not have a connection pool or a database for which all its connection pools have the option Shared Logon selected on the General tab of the Connection Pool dialog box. Identifies a database for which you cannot specify a user ID and password.

Table 7. Icons and Symbols

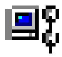
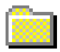
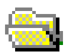
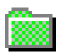
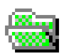
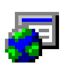
Icon or Symbol	What It Represents
	A connection pool in the physical layer.
	A collapsed catalog folder in the physical layer; a collapsed business model in the presentation and business model and mapping layers; and a collapsed sources folder within the logical table.
	An expanded catalog folder in 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 table registered as a Siebel event polling table.
	A logical table source in the business model and mapping layer.
	An alias table.
	An object that is a stored procedure call, as specified by the Object Type option in the General tab of the Physical Table dialog box.
	An object that has a Select (view) object type.
	A key for a physical or logical table.

Table 7. Icons and Symbols

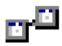
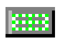
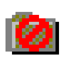
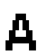
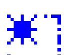
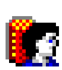
Icon or Symbol	What It Represents
	A foreign key for a physical or logical table in the Joins Manager.
	A physical or logical complex join in the Joins Manager.
	A dimension in the business model and mapping layer.
	A level in the business model and mapping layer.
	A physical or logical column.
	A logical column with an aggregation rule.
	A derived logical column.
	An invalid item.
	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.
	An item that contains an attribute definition.
	A new item (appears in online mode only). Can appear with other symbols, for example, to indicate a new item is an alias table.
	A system DSN ODBC entry. Appears in the Import dialog box.

Table 7. Icons and Symbols

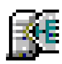
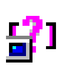
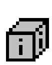
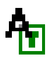
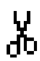
Icon or Symbol	What It Represents
	A measure definition.
	A user.
	A security group.
	An LDAP server.
	A group hierarchy.
	All users.
	An object that is checked out.
	A static repository variable.
	A dynamic repository variable.
	A system session variable.
	A nonsystem session variable.
	An initialization block.

Table 7. Icons and Symbols

Icon or Symbol	What It Represents
	A group association (appears only in the results display of the Query Repository dialog box).
	A level-to-level relationship (appears only in the results display of the Query Repository dialog box).
	A type privilege (appears only in the results display of the Query Repository dialog box).
	A query privilege (appears only in the results display of the Query Repository dialog box).
	A privilege package (appears only in the results display of the Query Repository dialog box).
	An object privilege (appears only in the results display of the Query Repository dialog box).
	An object that has been cut. Can appear with other symbols, for example, to indicate that a cut item is an alias table.

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 65

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. Enter 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, enter 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"
- "Using the Options Dialog Box—Cache Manager Tab" on page 72
- "Using the Options Dialog Box—Sort Objects Tab" on page 71
- "Using the Options Dialog Box—More Tab" on page 72

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
Edit object on creation	Automatically opens the dialog box for an object upon its creation.
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.

Option	Description
Show Time Wizard menu always	Keeps the Time Series Wizard menu active when working with a measure created by the wizard.
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. You can start the Time Series Wizard by selecting Tools > Time Series Wizard.
Show Calculation Wizard introduction page	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. 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.
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.

Option	Description
Show row count in physical view	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. 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 or XML Server databases.
Show Toolbar	Displays the toolbar.
Show Statusbar	Displays the status bar.

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.

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 scrolling speed of scrolling through the trees in various Administration Tool dialog boxes and to set the default window size for the join diagrams.

To set the scrolling speed slider

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.

To set the default window size for join diagrams

- In the Default diagram zoom drop-down list, choose a percentage for the window size.

  The default window size is 75 percent.

Setting Permissions for Repository Objects

You can use the Permissions dialog box to assign user and group permissions to an object. This 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 between 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:

- Database—General tab
- Connection Pool—General tab
- Catalog—General tab
- Physical Schema—General tab
- Physical Table—General tab
- Physical Column
- Dimension—General tab
- Logical Column—General tab
- Catalog Folder—General tab
- Presentation Table—General tab
- Presentation Column—General tab

TIP: You should not set permissions on physical layer objects except for connection pools. You should set permissions on presentation layer objects.

To add or edit permissions

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

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 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 being retrieved.

The Updating Row Counts window displays while row counts are being retrieved and stored. If you click Cancel, the retrieve process will continue for the table and its columns currently being processed, and then stop. Row counts appear for all tables and columns for which values were retrieved prior to the cancel operation.

NOTE: Row counts are not available for the following: items that have the Stored Procedure object type; XML data sources; XML Server databases; and data sources that do not support the CountDistinct function, like Microsoft Access and Excel.

To turn on the row count feature

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 the 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. Depending on the task you are performing, you can then perform relevant actions on the object, such as configuring permissions for the object and 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 a tree control listing all of the objects in the Presentation Layer, Business Model and Mapping Layer and the Physical Layer of a repository. The tabs at the bottom of the left pane allow you to select between the layers. Depending on the task you are performing, you will see only the layers that contain objects that can be manipulated in the context of the dialog box from which you entered the Browse dialog box.

The right pane of the Browse dialog box contains a facility 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.

To query for an object in the Browse dialog box

1. Select the object type from the Type drop-down list.
2. Enter the name of the object or a part of the name and the wild card character (*) in the Name field

   For example, to search for logical tables that have names beginning with the letter Q, select Logical Tables from the Type drop-down list, and then enter Q* 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

- Navigate to 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.

Setting Up and Working with a Repository 4

This section provides basic information about 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 and information about physical databases and other locations where the data is stored. It defines the logical business models which are presented to users.

Components of a Repository in the Administration Tool

The Siebel Analytics Server 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

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 be present 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 "Working with Connection Pools" on page 121.

Business Model and Mapping Layer

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

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

You can change the names and set the permissions 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."

Presentation Layer

The actual presentation of a business model to users is represented by the Presentation layer. 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 also display column names that are different from the column names in the Business Model and Mapping layer.

The Presentation layer is a good location to set permissions. You can set permissions for users to grant or deny them access to individual business models or columns.

The Presentation layer also presents localized language translations.

Security

Siebel Analytics Server Security is where you set up users, groups, access privileges, and query privileges. Full group-based security provides rapid administration of user permissions. For more information, see "Security Manager" on page 315.

Context-Sensitive Menus

Most of the objects in a repository have context-sensitive menus associated with them. The context-sensitive menus contain commands specific to the object that is selected or commands specific to the location of the cursor.

To access a context sensitive menu, select an object and then right-click. A menu specific to that object appears. You can highlight a specific menu item and press the F1 key to view the online help for that item.

The Steps to Setting Up a Repository

When you understand your business model, you can begin to set up a repository. This section describes the steps for setting up a repository using the Siebel Analytics Server Administration Tool.

Table 8 lists the tasks to set up a repository and the page number where each task is described.

Table 8. Repository Setup Checklist

Step #	Task
Step 1	"Creating a New Repository" on page 83
Step 2	"Importing the Physical Schemas" on page 83
Step 3	"Create and Configure the Connection Pool" on page 85
Step 4	"Define Join Conditions on the Physical Schema" on page 85
Step 5	"Create the Business Model" on page 86
Step 6	"Creating the Presentation Layer" on page 86
Step 7	"Save Repository, Check Consistency, and Correct Any Errors" on page 88
Step 8	"Add an Entry in the NQSConfig.INI File" on page 90
Step 9	"Create the Data Source" on page 91
Step 10	"Start the Siebel Analytics Server" on page 91
Step 11	"Test and Refine the Repository" on page 91
Step 12	"Publish to User Community" on page 91

Creating a New Repository

The steps in the following procedure explain how to create a new repository.

To create a new repository

1. Select File > New from the Administration Tool menu or click the New button in the tool bar.
2. Enter 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 8 on page 82, 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.

Dragging and Dropping an Existing Repository

Once you have created a repository, you can also drag and drop it from one instance of the Administration Tool to another instance. This allows you, for example, to develop and test a different business model based upon the same Physical layer, and then combine it with a production repository. Other advantages of using drag-and-drop in this manner include faster repository development and development without any impact on the production system.

Importing the Physical Schemas

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, but 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.

The first step in creating the Physical layer is to import schemas or portions of schemas from existing data sources.

NOTE: Physical metadata imports have to connect using an ODBC connection; native database gateways for metadata import are only supported for DB2 (using DB2 CLI or DB2 CLI Unicode) and XML. To import XML data using the Siebel Analytics Server XML gateway, see "Siebel Analytics Server XML Gateway Example" on page 348.

To import a schema

1. In the Administration Tool, select File > Import.
2. Select a connection type and a data source to import from, and then click OK.
3. If a logon screen appears, enter a valid user ID and password for the data source, and then click OK.

   The Import dialog box appears.
4. Check the type of metadata you want to import.

   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.

   The default varies based on your data source. For most databases, the default is to import tables, primary keys, and foreign keys. Note that you can also import database views, aliases, synonyms, and system tables. Import these other objects only if you want the Siebel Analytics Server to generate queries against them.

5 Select the tables and columns you want to import.

- To select multiple objects, select the first object and then control-click to select other objects.
- To select multiple adjacent objects, select the first object and then shift-click the last adjacent object.

6 When the objects are selected, either click Import or drag and drop the objects into the Physical layer. This imports the metadata for the objects into the Physical layer.

NOTE: When you import physical tables, be careful to limit the import only to those objects that are likely to be used as sources of data in the business models you create. Importing too many extraneous objects will complicate your job as an administrator, increase the size of the repository, and potentially affect the performance of the Administration Tool.

Create and Configure the Connection Pool

The *connection pool* is an object in the Physical layer that describes access to the data source. When you import a schema using a data source as described in the previous step, the connection pool is created automatically. You can create other connection pools using the Connection Pool dialog box. For information about creating and configuring connection pools, see "Working with Connection Pools" on page 121.

Define Join Conditions on the Physical Schema

You need to define the joins that express relationships between tables in the Physical layer of the repository. If the imported database joins over primary key-foreign key relationships and the keys were imported into the repository, the join conditions are set up automatically. For information about defining physical join conditions, see "Defining Physical Joins in the Physical Table Diagram" on page 143 and "Defining Physical Joins with the Joins Manager" on page 141.

Create the Business Model

The business model contains the logical schema, and it forms the basis of the users' view of the data. You create one or more business models and create logical tables and columns in each business model. To map objects in the Business Model and Mapping layer to sources in the Physical layer, you can drag and drop Physical layer objects into the Business Model and Mapping layer.

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

- "Working with Business Models" on page 149
- "Creating and Editing Logical Tables" on page 151
- "Creating a Dimension" on page 166
- "Defining Physical to Logical Table Mapping" on page 162
- "Defining Content of Sources" on page 163
- "Setting Default Levels of Aggregation for Measure Columns" on page 157
- "Defining Logical Joins with the Joins Manager" on page 176
- "Defining Logical Joins with the Business Model Diagram" on page 178

Creating the Presentation Layer

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 for creating the Presentation layer.

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 15, “Security.”

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.

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.

Save Repository, Check Consistency, and Correct Any Errors

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. See "Checking the Consistency of a Repository or a Business Model" on page 66.

Add an Entry in the NQSConfig.INI File

Once you have built 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 ;
   ```

   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 users' client applications to connect to the new repository, each user needs to define a data source using an ODBC driver. (Remember that 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 12, "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 NQSConfig.INI file are loaded and become available for querying. For detailed information on starting the server, see "Starting the Siebel Analytics Server" on page 217.

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 enter 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.

Synchronizing Repositories

Use the Administration Tool's Repository Import Wizard to synchronize differences between repositories. This allows you to update and synchronize an open repository's catalogs, users, groups, and variables with those of another repository. When you import objects from one repository into another, the repository from which you are importing must be consistent.

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

To synchronize and update a repository

1 In the Administration Tool, select File > Import from Repository.

2 In the Repository Import Wizard, select the repository type you want to update from, and then click Next.

   The listed repositories belong to one of the following categories:

   - Repositories that are opened in other instances of the Administration Tool on the same machine. These are identified in the Type column as Siebel Analytics Administration Tool.
   - Data Source Names (DSNs) for repositories. These are identified in the Type column as DSN.
   - Repositories in the Repository directory, located in the Siebel Analytics Server software installation folder. These are identified in the Type column as Repository.

   NOTE: You can update from only one repository at a time.

   ---

3 Perform one of the following tasks:

- If you chose an online update source of type DSN, the Open Online Repository dialog box appears. Enter your Siebel Analytics Server administrator user ID and password. To immediately load all objects when opening the repository, select the option Load All Objects.

  This causes all repository objects to be loaded on initial connection rather than when being selected for work. This may slow the initial connection time, but it speeds the execution of other tasks in the Administration Tool.

- If you chose an offline update source of type Repository, the Open Offline dialog box appears. Enter your Siebel Analytics Server administrator user ID and password.

- If you chose an update data source of type Siebel Analytics Administration Tool, you will not be prompted for logon information.

4 From the drop-down list at the top of the Repository Import Wizard—Objects to Update dialog box, select the category of objects you want to update.

  Objects appear in the left pane of the dialog box. Depending on the category of objects you selected, different buttons allow you to specify how you want to add objects.

5 Do one of the following:

- When you select the Catalogs category, the presentation catalogs appear in the left pane and the Add with Children button becomes active. Click the button to add the presentation catalogs to the right pane. Presentation catalogs are always added with all their child objects, and all associated objects, from the Presentation layer to the Physical layer, will be updated and synchronized.

- When you select the Groups category, you can click Add to update and synchronize only the group, or the Add with Children button to update and synchronize the group and all the groups that are below it in the group hierarchy, or the Add with Parents button to update and synchronize the group and all groups that are above it in the group hierarchy. (You can view group hierarchies from the Security Manager.)

- When you select the Users category, the buttons that will become active depend on the user selected in the left pane. For example, if the highlighted user belongs to a group, the Add with Parents button will become active. You can click the Add button to update and synchronize only the user, or the Add with Parents button to update and synchronize the user and the user's parent group. (You can view information about group membership from the Security Manager.)
- When you select the Variables category, the defined system and session variables appear in the left pane. You can click the Add button to update and synchronize only the variable, or the Add with Parents button to update and synchronize the variable and any parent variables on which it depends.

You can repeat this step until you have added all the object you want to synchronize and update.

If the objects to be updated and synchronized need to be checked out, the Check Out Objects screen appears.

6 Click Next to continue.

The Finish screen displays the repository objects that will be updated and synchronized.

7 Click Finish to complete the update and synchronization process.

The Repository Import Wizard closes automatically. If the Administration Tool is open in online mode, you will be prompted to check in the changes before closing the repository.

Managing Repository Metadata

You can use the Query Repository dialog box to do the following:

- Examine and update the internal structure of the repository. For example, you can search for objects in the repository based on name, type (such as Catalog, Complex Join, Key, and Level), 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.
- View reports that show such items as all tables mapped to a logical source, all references to a particular physical column, the content filters for logical sources, dimensional hierarchies, 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. The report results are generated using Universal Database Markup Language (UDML) and can be viewed online or copied and pasted into a program such as Windows Notepad. The results can also be saved to a file in comma-separated value (.csv) format. This file could then be imported into or opened in a program such as Microsoft Access to create a table of the results, making them available for querying through Siebel Answers.

To access the Query Repository dialog box

- In the Administration Tool, select Tools > Query Repository.

To search for objects based on name

- In the Query Repository dialog box, enter the text to search for in the Name field, and then click Query.

  You can use an asterisk (*) wildcard character to specify any characters. The wildcard character must be the last character in the search string. For example, to search for all objects beginning with the letter 'o', enter the following:

  o*

NOTE: Searches are not case sensitive.

To narrow your search to a particular type of object

- Select the type of object from the Type drop-down list, and then click Query.

To display fully qualified names in the Results pane

- Select the option Show Qualified Name.

  This functionality is useful in seeing which physical table a particular physical column belongs to.

To see the parent of an object

- Select an object in the results list, and then click Parent.

  The Parent Hierarchy dialog box appears for the selected object. (If the object does not have a parent, a message displays instead). Using this feature, you can display the full parent hierarchy for any object that has a parent.

- From the Parent Hierarchy dialog, you can edit and delete objects:
  - Click Edit to open the appropriate dialog box for the type of object you are editing.
  - Click Delete to delete the selected object. Any child objects of the selected object will also be deleted.

To create a new object

1. In the Type drop-down list, select the type of object you want to create.

   The New button becomes active.

2. Click New to open the appropriate dialog box for the type of object you want to create.

To edit an object in the Results list

- Select the object, and then click Edit.

  The appropriate dialog box opens for the object.

NOTE: Not all repository objects can be edited from the Results list; examples include privilege objects and user database sign on objects, among others. If an object cannot be edited from the Results list, the Edit button will not be available.

To report on an object and view the results online

1. Select the object, and then click UDML.

   The UDML window appears.

2. To copy the results to the Windows clipboard, click Copy.
3. Open a program such as Windows Notepad and paste the results.

NOTE: You can also paste UDML into the Execute UDML utility window and use it to create an object in 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 so far.

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 Repositories" on page 103.

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 64.

2. Select File > Compare from the toolbar.

   The Select Original Repository dialog box appears.

3. Select the repository you want to compare with the current repository (that is, the repository you opened in Step 1).

   The Open Offline dialog box appears.

4 Enter the password and click OK.

The Compare Repositories dialog box displays the differences between the two repositories.

Values in the Change column indicate the following:

Value	Description
Created	Object was created in the current repository and does not exist in the original repository.
Modified	Object exists in the original repository but has been modified in the current repository.
Deleted	Object exists in the original repository but has been deleted from the current repository.

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

Button	Functionality
Select	Allows you to select a repository to compare with the current repository.
View 1	Opens deleted objects in read-only mode.
Edit 2	Opens created objects for editing.
UDML 1	Displays the Universal Data Modeling Language (UDML) information for objects in the original repository.
UDML 2	Displays the UDML information for objects in the current repository.
Diff	Displays the differences between the current repository and the original repository.
Save	Saves a list of the differences between the current repository and the original repository.
Mark	Marks the object you select. Boxes appear around created and modified objects. To remove marks, enable the Turn Off Compare 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 Repositories

This section is for organizations that are using the Siebel Analytics applications repository (SiebelAnalytics.rpd). It explains how to use the Merge Repository feature of the Administration Tool to merge your customizations of prior releases of the repository 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 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.

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.

To merge versions of Siebel Analytics Repository

1. Rename the original repository (the repository you received with the previous version of Siebel Analytics) SiebelAnalytics.Prev.rpd.
2. Rename the modified repository SiebelAnalyticsCustom.rpd.
3. Rename the current repository SiebelAnalytics.Current.rpd.
4. In the Administration Tool, open SiebelAnalytics.Current.rpd in offline mode.
5. Select File > Merge from the toolbar.

   The Select Original Repository dialog box appears.
6. Select SiebelAnalytics.Prev.rpd.

7 Enter the password and click OK.

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

Values in the Change column indicate the following:

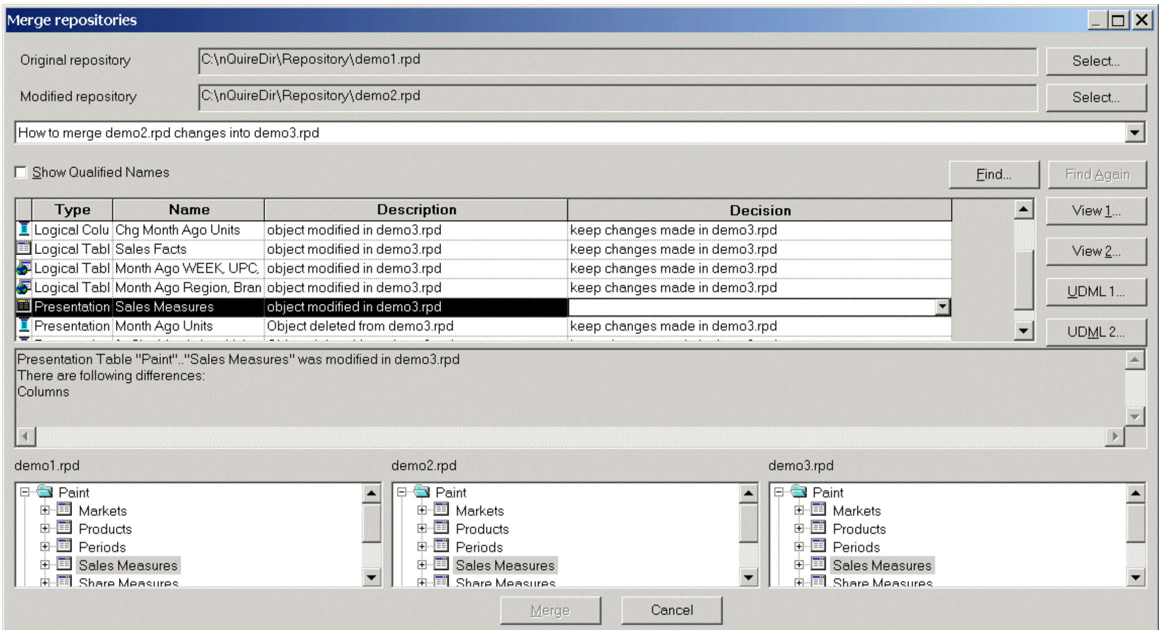
Value	Description
Created	Object was created in the current repository and does not exist in the original repository.
Modified	Object exists in the original repository but has been modified in the current repository.
Deleted	Object exists in the original repository but has been deleted from the current repository.

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

Button	Functionality
Select	Allows you to select a repository to compare with the current repository.
View 1	Opens deleted objects in read-only mode.
Edit 2	Opens created objects for editing.
UDML 1	Displays the Universal Data Modeling Language (UDML) information for objects in the original repository.
UDML 2	Displays the UDML information for objects in the current repository.
Diff	Displays the differences between the current repository and the original repository.
Save	Saves a list of the differences between the current repository and the original repository.
Mark	Marks the object you select. Boxes appear around created and modified objects. To remove marks, enable the Turn Off Compare Mode.

8 In the Modified Repository field, select SiebelAnalyticsCustom.rpd.

The Merge Repositories dialog box appears.



The names of the original and modified repositories appear at the top. The Select buttons to the right of the names allow you to select another repository.

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.

9 Select the option How to merge SiebelAnalyticsCustom.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 following table shows the possible repository changes and the corresponding decision options.

Description	Decision Options
Object(s) added to <current>	■ Keep changes made in <current> (default) ■ Remove objects created in <current>
Object(s) added to <modified>	■ Add these objects (default) ■ Don’t add these objects
Object(s) deleted from <current>	■ Keep changes made in <current> (default) ■ Add back objects not deleted in <modified>
Object(s) deleted from <modified>	■ Add these objects (default) ■ Don’t add these objects
Object(s) modified in <modified>	■ Make changes modified in <modified> (default) ■ Don’t modify this object
Object(s) modified in <current>	■ Keep changes made in <current> (default)
Object(s) deleted from both <current> and <modified>	■ Keep changes made in both repositories (default)
Object(s) modified in both repositories; there are differences	■ Keep changes made in <current> ■ Make changes made in <modified>
Object(s) modified in both repositories; there are no differences	■ Keep changes made in both repositories (default)

Description	Decision Options
Object(s) deleted from <current>; some also from <modified>	■ Keep changes made in <current> ■ Add back objects not deleted in <modified>
Object(s) deleted from <modified>; some also from <current>	■ Keep changes made in <current> ■ Remove objects deleted in <modified>

The read-only text box below the grid provides detailed information about the repository change for the object you select in the grid.

10 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.

11 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.

12 Click Merge.

A message will alert you when the merge is complete.

13 Save the current repository.

Working in a Repository's Presentation Layer 5

This section provides instructions for using the Administration Tool to create and edit objects in the Presentation layer of a repository.

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 112
- "Working with Presentation Columns" on page 113
- "Using the Alias Tab of Presentation Layer Dialog Boxes" on page 115

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 9.

Table 9. 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, enter 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 73.

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

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) Enter 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.

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.

Presentation Tables Dialog Box

The Presentation Tables dialog box has three tabs: General, Columns, and Aliases. The functionality provided in each tab is described in Table 10.

Table 10. 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 73.

4. (Optional) Enter a description of the table.

To delete a presentation column

1. Click the Columns tab.
2. Select the column you want to delete.
3. Click Remove, or press the Delete key.

To reorder a presentation column

1. Click the Columns tab.
2. Select the column you want to reorder.
3. Use the drag-and-drop feature to reposition the column, or click the Up and Down buttons.

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.

Presentation Column Dialog Box

The Presentation Column dialog box has two tabs: General and Aliases. The functionality provided in each tab is described in Table 11.

Table 11. 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 Enter a name for the column.

b Clear the Use Logical Column check box.

c Clear the Custom display name 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) Enter a description of the presentation column.

8 To define any aliases for the logical column, click the Aliases tab.

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 Enter 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.

Working in a Repository's Physical Layer 6

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

Physical Layer Objects

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 multi-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.

This section includes the following topics:

- "Working with Databases" on page 118
- "Working with Connection Pools" on page 121
- "Working with Catalog Folders" on page 128
- "Working with Schema Folders" on page 130
- "Working with Physical Tables" on page 131
- "Specifying a Primary Key" on page 136
- "Working with Physical Columns" on page 136

Working with Databases

The Physical layer of the Administration Tool contains one or more databases. The databases can be of the same or different RDBMS varieties as well as XML data sources. For information about supported databases, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

You can use the Database dialog box to perform the following tasks:

- "Creating or Editing a Database Object"
- "Specifying Types of Queries Sent to a Database" on page 119

Creating or Editing a Database Object

Use the General tab of the Database dialog box to create or edit a database object in the Physical layer of the Administration Tool.

To create or edit a database object

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

   The Database dialog box appears.

2. In the General tab, do the following:

   a. Enter a name for the database.

   b. In the Data source definition area, use the drop-down list to specify the database type.

   NOTE: The database type is set automatically when the metadata is imported to the Physical layer. During the metadata import process, some ODBC drivers provide the Siebel Analytics Server with the database type. If the server cannot determine the database type, the appropriate ODBC type is assigned to the database object.

c (Optional) Enter a description of the database.

d (Optional) Click Permissions to open the Permissions dialog box, where you can assign user or group permissions to the database.

For information about assigning permissions to a database, see "Setting Permissions for Repository Objects" on page 73.

3 (Optional) Click the Features tab to examine the types of queries supported by the database type specified in the previous step.

Specifying Types of Queries Sent to a Database

The Features tab of the Database dialog box contains a Feature table, which lists the SQL features supported by the data source. When a database type is specified in the General tab of the Database dialog box, the Feature table is automatically populated with default settings, which indicate the SQL features the server uses with this data source.

You can change the default settings. 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 query types that the database does not support, your query may return errors and unexpected results. Also, disabling query types that are supported can result in the server issuing less efficient SQL to the database.

To specify types of queries sent to a database

1 In the Physical layer of the Administration Tool, double-click the database for which you want to specify query types.

2 In the Database dialog box, select the Features tab.

The Feature table includes two check boxes: Default and Value. The Default check box is automatically selected for the query types that are supported for the database type specified in the General tab. The Value check box allows you to enable query types other than the default query types.

3 Do one of the following:

- Select the Value check box to enable a query type.
- Clear the Value check box to disable a query type.

To locate a query type

1 In the Features tab of the Database dialog box, click Find.

2 Enter a string to search for, and then click OK.

To restore the default entries for the selected database type

- In the Features tab of the Database dialog box, click Revert to Defaults.

Changing Feature Table Entries Using Ask DBMS

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

The Ask DBMS button is not available if you are using an XML data source.

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

To change Feature table entries using Ask DBMS

1 In the Features tab of the Database dialog box, click Ask DBMS.

2 In the Browse dialog box, select a connection pool with which to query the data source.

Working with Connection Pools

A connection pool contains information about the connection between the Siebel Analytics Server and a 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.

For each connection pool, you must specify the maximum number of concurrent connections allowed. Once 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. Therefore, raising the number of connections increases Siebel Analytics Server memory usage.

This section includes the following topics:

- "Creating or Editing a Connection Pool" on page 121
- "Setting Properties for an XML Data Source" on page 135

Creating or Editing a Connection Pool

Use the General tab of the Connection Pool dialog box to create or edit a connection pool in the Physical layer of the Administration Tool. You must create a database object before you create a connection pool.

To create or edit a connection pool

1 In the Physical layer of the Administration Tool, do one of the following:

- To create a connection pool, right-click a database and select New Object > Connection Pool from the shortcut menu.
- To edit an existing connection pool, double-click the connection pool icon for the connection pool you want to edit.

The Connection Pool dialog box appears. Some of the fields are described in the following table.

Field	Comments
Name	The name for the connection pool. If you do not enter a name, the Administration Tool generates a name.
Permissions	Click to access the Permissions dialog box. You can use the Permissions dialog box to specify access permissions for individual users or groups. 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.
Maximum connections	The maximum number of connections allowed for this connection pool.

Field	Comments
Require fully qualified table names	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. 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 *foo* first looks for that table in the master database, and then looks for it in the specified database. If the table named *foo* exists in the master database, that table is queried, not the table named *foo* in the specified database.
Data source name	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.

Field	Comments
Shared logon	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 ID and password. If this option is selected, then all connections to the database that use the connection pool will use the user ID and password specified in the connection pool, even if the Siebel Analytics user has specified a database user ID and password in the DSN (or in the Siebel user configuration). 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.
Enable connection pooling	Allows a single database connection to remain open for the specified time for use by future query requests. Connection pooling saves the overhead of opening and closing a new connection for every query. If you do not select this option, each query sent to the database opens a new connection.
Timeout (Minutes)	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. 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.

Field	Comments
Execute queries asynchronously	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.
Execute on Connect	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.
Isolation level	For ODBC and DB2 gateways, sets the transaction isolation level on each connection handle to the back-end database.

2 Enter a name for the connection pool.

3 Select a call interface.

NOTE: If the call interface is XML, the XML tab becomes active and options that do not apply to XML data sources will not be available.

4 Specify the maximum number of connections for the pool.

5 Select the option Require fully qualified table names if the database requires it.

6 Select or enter the data source name that the connection pool will access.

The drop-down list shows the User and System DSNs configured on your system.

7. If the connection pool uses a shared logon, select the option Shared logon and specify a user ID and password for the source.
8. To allow a single database connection to remain open for a specified number of minutes for use by future query requests, select the option Enable connection pooling.

   NOTE: If you do not select this option, each query sent to the database will open a new connection.

   ---

9. Enter the number of minutes that a connection to the data source will remain open after a request completes in the Timeout text box.
10. Select the option Execute queries asynchronously for data sources that support asynchronous queries.
11. To specify a command to be executed each time a connection is made to the database, enter the command string in the Execute on Connect text box.
12. Select an isolation level, if applicable.
13. (Optional) Enter a description of the connection pool.
14. Click Permissions to open the Permissions dialog box, where you can assign user or group permissions to the connection pool.

    For information about assigning permissions to a connection pool, see "Setting Permissions for Repository Objects" on page 73.
15. Complete the XML tab if you specified XML as the data source.

Setting 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.

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

To set connection properties

1 (Optional) If you specified a URL to access the data source, set the URL refresh interval:

   a Select a value from the drop-down list (Infinite, Days, Hours, Minutes or Seconds).

   b Specify a whole number as the numeric portion of the interval.

   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.

2 (Optional) If you specified a URL to access the data source, set the URL loading time-out:

   a Select a value from the drop-down list (Infinite, Days, Hours, Minutes or Seconds).

   b Specify a whole number as the numeric portion of the of the interval.

   The default time-out interval for queries (URL loading time-out) is 15 minutes.

To specify query output format settings

1 (Optional) For an XSLT file, enter 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, enter the XPath expression in the XPath Expression field, for example, //XML, or use the Browse button.

Working with Catalog Folders

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

You can use the Catalog dialog box to perform the following tasks:

- "Creating or Editing a Catalog"
- "Using a Variable to Specify the Name of a Catalog or Schema" on page 129

Creating or Editing 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 and select New Object > Catalog from the pop-up menu.

   The Catalog dialog box appears.

2. Enter a name for the catalog.
3. (Optional) Click Permissions to open the Permissions dialog box, where you can assign user or group permissions to the catalog.

   For information about assigning permissions to a catalog, see "Setting Permissions for Repository Objects" on page 73.

4. Enter a description for the catalog, 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 unassign a session variable in the Dynamic Name tab

- Click Clear to unassign 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.

Working with 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 can use the Schema dialog box to perform the following tasks:

- "Creating and Editing a Schema"
- "Using a Variable to Specify the Name of a Schema"

Creating and Editing a Schema

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

To create a schema object

1. In the Physical layer, select the database for which you want to create a schema.
2. Right-click and select New Object > Schema.

   The Schema dialog box appears.
3. Enter a name for the schema.
4. (Optional) Click Permissions to open the Permissions dialog box, where you can assign user or group permissions to the catalog.

   For information about assigning permissions to a schema, see "Setting Permissions for Repository Objects" on page 73.
5. Enter a description for the schema, and then click OK.

Using a Variable to Specify the Name of a Schema

Use the Dynamic Name tab to use a variable to specify the names of schema objects. The Dynamic Name tab in the Schema dialog box provides the same functionality as the Dynamic Name tab in the Catalog dialog box. For information about using this tab, see "Using a Variable to Specify the Name of a Catalog or Schema" on page 129.

Working with 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.

Object Types

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 212. Creating virtual tables can provide the Siebel Analytics Server and the underlying databases with the proper metadata to perform some advanced query requests.

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 12 provides a description of the available object types.

Table 12. Object Type Descriptions

Object Type	Description
None	Specifies that the physical table object represents a physical table.
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 enter the alias name. The alias you enter must be a valid physical table in the database specified in the connection pool.

Table 12. Object Type Descriptions

Object Type	Description
Stored Proc	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 enter the stored procedure. Requests for this table will call the stored procedure. 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 database's default configuration will be executed.
Select	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 enter the select statement. Requests for this table will execute the Select statement. 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. If a table with a Select object type is to be joined to any other physical table, you need to include the appropriate foreign keys in the Select statement. 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 database's default configuration will be executed.

Using the Physical Table Dialog Box

In the Physical Table dialog box you can define general properties, columns, a primary key, and foreign keys. If the table definitions are imported together with their columns and key information, the properties are configured automatically (assuming that the keys are defined in the database).

You can use the Physical Table dialog box to perform the following tasks:

- "Creating or Editing a Physical Table"
- "Using the Columns, Keys, and Foreign Keys Tabs" on page 135
- "Setting Properties for an XML Data Source" on page 135

Creating or Editing 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, do one of the following:
   - To create a physical table, right-click a database and select New Object > Physical Table from the shortcut menu.
   - To edit an existing physical table, double-click the physical table icon for the connection pool you want to edit.
2. Enter a name for the table.

   The name of the table and its associated path are displayed in the Object Name field. When XML data is being used as a data source, this field displays the fully qualified name of a table used in an XML document.

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 physical table, see "Setting Permissions for Repository Objects" on page 73.

TIP: You should not set permissions on physical layer objects except for connection pools. You should set permissions on presentation layer objects.

---

4 To include the table in the Siebel Analytics Server query cache:

   a Select the option Make table cacheable.

   This causes the Cache persistence time settings to become active. These settings 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. In general, you should check this option for most tables.

   b To specify in minutes or seconds how long the entries should persist, select (minutes) or (seconds) from the drop-down list on the right, and type a whole number into the field on the left.

   The default value is Infinite, which means that cache entries do not automatically expire. 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, still result in entries being removed from the cache.

   For information about the cache polling table, see "Troubleshooting Problems with an Event Polling Table" on page 262.

NOTE: 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.

---

5 Select an Object Type from the drop-down list, and, if necessary, enter the appropriate text in the text pane.

See Table 12 on page 131 for a description of the available object types.

6 (Optional) If you selected an object type of None or Alias, you can enter a database hint.

For information about database hints, see "Using Database Hints" on page 145.

7 (Optional) Type a description of the table.

Using the Columns, Keys, and Foreign Keys Tabs

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

These are 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 object you select.

Setting 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, the properties you set in the Physical Table dialog box override the corresponding settings in the Connection Pool dialog box. For instructions on setting properties in this tab, see "Setting Properties for an XML Data Source" on page 126.

Specifying a Primary Key

Use the Physical Key dialog box to specify the column or columns that define the primary key of the physical table. You can access the Physical Key dialog box from the Keys tab of the Physical Table dialog box.

To specify a primary key

1. In the Physical Key dialog box, enter a name for the key.
2. Select the check box for the column that defines the primary key of the physical table.
3. 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.
4. (Optional) Enter a description for the key.

Working with Physical Columns

Each table in the Physical layer of the Administration Tool has one or more physical columns. The properties of the column are set automatically when the column is imported.

Use the Physical Column dialog box to create or edit a physical column.

To create or edit a physical column

1. In the Physical layer of the Administration Tool, do one of the following:
   - To create a physical column, right-click a physical table and select New Physical Column from the shortcut menu.
   - To edit an existing physical table, double-click the physical column icon for the connection pool you want to edit.

2 In the Physical Column dialog box, enter 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 Click the Permissions button to open the Permissions dialog box, where you can assign user or group permissions to the physical column.

   For information about assigning permissions to a physical column, see "Setting Permissions for Repository Objects" on page 73.

4 In the Type field, select a datatype for the physical column.

5 If applicable, specify the length of the datatype.

6 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.

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

About Physical Joins

All valid physical joins need to be configured in the Physical layer of the Administration Tool. When you import a physical schema and import keys, the primary key-foreign key joins are automatically defined. Any other joins within each database or between databases have to be explicitly defined.

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 by a table in one database that logically joins to a table in another database. You need to specify multi-database joins in order to combine the data from different databases. Use the Physical Diagram editor 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 using the Physical Diagram editor, see "Defining Physical Joins in the Physical Table Diagram" on page 143.

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 11 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).

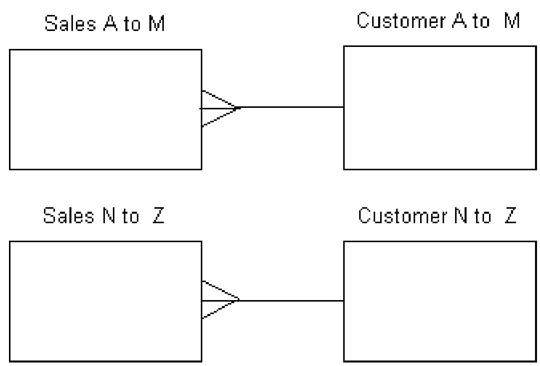


Figure 11. Fragmented Tables Example

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 12.

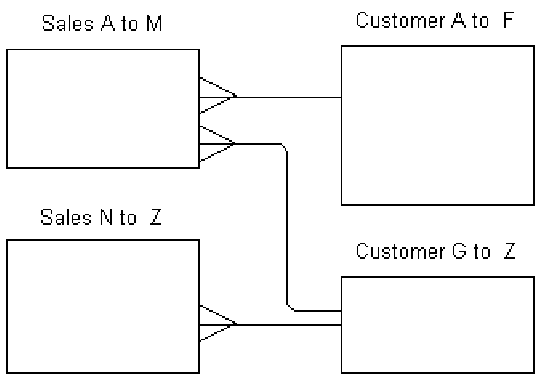


Figure 12. Joins for Fragmented Tables Example

TIP: Avoid adding join conditions where they are not necessary (for example, between Sales A to M and Customer N to Z in Figure 11). 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 Table Diagram" on page 143 and "Using the Columns, Keys, and Foreign Keys Tabs" on page 135.

Complex Joins

Complex joins are joins over nonforeign key and primary key columns; that is, the relationship between the tables is arbitrary. 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. A complex join in the Business Model and Mapping layer acts as a placeholder when the business model has no foreign keys but requires a physical join to be created with expression and column information. A complex join in the Physical layer does not require a matching join in the Business Model and Mapping layer.

You can create physical and logical complex joins using the Joins Manager or using the Physical or Logical Table Diagram. For more information, see "Defining Physical Joins with the Joins Manager" on page 141 and "Defining Physical Joins in the Physical Table Diagram" on page 143.

Defining Physical Joins with the Joins Manager

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

NOTE: Primary key and foreign key joins must be created in the Keys tab and Foreign Keys tab of the Physical Table dialog box. See also "Defining Physical Joins in the Physical Table Diagram" on page 143.

To create a physical foreign key or complex join

1. In the Administration Tool tool bar, select Manage > Joins.

   The Joins Manager dialog box appears.

2 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.

  The Physical Foreign Key dialog box appears.

3 Enter 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 145.

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 Table Diagram

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

To display the Physical Table Diagram

- Right-click on a database in the Physical layer of the Administration Tool, and select one of the following options from the shortcut menu:

Physical Diagram Menu	Description
Selected object(s) only	Displays the selected objects only. Joins are displayed only if they exist between the objects selected.
Object(s) and direct joins	Displays the selected objects, as well as any tables that join to the selected objects.
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 of the objects in a schema are related, then using this option will diagram every table, even if only one table is selected.

You can add other tables to the diagram by selecting other physical tables, schemas, or databases, and executing the Physical Table Diagram command from the right-click menu.

To define a foreign key join or 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.

   The Physical Table Diagram window appears.

2. To add another table to the diagram window, select the table you want to add (with the diagram window still open) and execute one of the Physical Diagram commands from the right-click menu.

   Repeat this process until all the tables you are interested in appear in the diagram window.

3 Click one of the following buttons on the Administration Tool tool bar:
   - New foreign key
   - New complex join
4 With this button selected, move the cursor to the first table in the join (the one of the one-to-many join).
5 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 Physical Foreign Key or Physical Join dialog box appears.
6 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 180.

---

7 If appropriate, specify a database hint.

   For information about database hints, see "Using Database Hints" on page 145.
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 apply the selections.

   The join link appears in the diagram window with the crow's feet pointing to the many part of the one-to-many join.

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 7.5 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 enter 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 enter 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 enter 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 Enter 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 enter SQL comment markers (/* or --) when you enter the text of the hint. The Siebel Analytics Server inserts the comment markers when the hint is executed.

Working in a Repository's Business Model and Mapping Layer 7

This section provides instructions for using the Administration Tool to create and edit objects in the Business Model and Mapping layer of a repository.

Business Model and Mapping Layer Objects

The Business Model and Mapping layer of the Administration Tool defines the business model of the data and specifies the mapping between the business model and the physical schemas where the data is actually stored.

The section provides instructions for using the Administration Tool's Business Model and Mapping layer dialog boxes to create and edit logical repository objects.

This section includes the following topics:

- "Working with Business Models"
- "Working with Logical Tables" on page 150
- "Working with Logical Columns" on page 155
- "Working with Logical Table Sources (Mappings)" on page 160
- "Working with Dimensions" on page 165

Working with Business Models

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 logical to physical table mappings for the business model.

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.

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) Enter a description of the business model, and then click OK.

Working with Logical Tables

Logical tables are displayed in the Business Model and Mapping layer. Each table is displayed with the same icon, regardless of whether it behaves as a fact table, a dimension table, or as part of a snowflaked dimension in the schema. Each logical table has one or more logical columns and one or more logical table sources associated with it.

You can use the Logical Table dialog box to change the name, reorder the sources, configure the keys (primary and foreign), and modify permissions.

This section includes the following topics:

- "Creating and Editing Logical Tables" on page 151
- "Using the Logical Table Dialog Box—Key Tab" on page 153
- "Using the Logical Table Dialog Box—Sources Tab" on page 154
- "Using the Logical Table Dialog Box—Foreign Keys Tab" on page 155

Creating and Editing Logical Tables

There are two ways to create logical tables:

- Create them explicitly in the Business Model and Mapping layer.
- Drag and drop a physical table from the Physical layer onto a business model 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, all of the columns belonging to the table are included with it. You can then modify or delete the resulting logical objects in the Business Model and Mapping layer. Drag and drop operations are usually the fastest method for creating objects in the Business Model and Mapping layer. When you have dragged objects into the Business Model and Mapping layer, you can modify them in any way necessary without affecting the Physical layer.

To create a logical table explicitly

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, enter 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 51.

4. (Optional) Enter a description of the table.

To create a logical table by dragging and dropping

1. Select one or more objects in the Physical layer.
2. Drag the objects to a business model in the Business Model and Mapping layer.

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

NOTE: 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 physical joins. However, this occurs only when the tables that are being dragged include the table with the foreign keys. Once you create logical tables, if additional physical tables are dragged to the logical business model, the logical links between the newly dragged tables and the existing logical tables need to be created manually. For more information, see "Defining Logical Joins with the Joins Manager" on page 176 and "Defining Logical Joins with the Business Model Diagram" on page 178.

To add a logical column to the table

1. Click the New button to open the General tab of the Logical Column dialog box.
2. Complete the dialog box, and then click OK.

   Information about the column's name, type, length, and whether it is nullable is displayed in the column window.

To edit a logical column

1. In the Logical Columns pane, select the column you want to edit.
2. Click Edit.

   The Logical Column dialog box appears.
3. Make your changes, and then click OK.

To delete a logical column

1. In the Logical Columns list, select the column you want to delete.
2. Click Remove, or press the Delete key.

To reorder a logical column

1. In the Logical Columns list, select the column you want to reorder.
2. Use the drag-and-drop feature to reposition the column, or click the Up and Down buttons.

Using the Logical Table Dialog Box—Key Tab

Use the Keys tab to identify the primary key table.

To specify a key

1. Open the Logical Key dialog box by selecting the Key tab, and then clicking New.

   The Logical Key dialog box appears.

2. Enter a name for the key.
3. 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) Enter a description for the key.
6. Click OK to return to the Logical Table dialog box.

   The key is added to the Primary Key drop-down list in the Logical Key dialog box.

To edit a key

1. Select the key you want to edit and double-click it to open the Logical Key dialog box.
2. Make your changes.
3. Click OK to return to the Logical Table dialog box.

To delete a key

- Select the key you want to delete, and then click Delete.

Using the Logical Table Dialog Box—Sources Tab

Use the Sources tab to reorder the logical table sources in the Administration Tool workspace. You can also add a new source, and edit or delete an existing source.

To add a new logical table source

1. Open the Logical Table Source dialog box by selecting the Sources tab, and then clicking Add.
2. Complete the Logical Table Source dialog box.

To edit an existing logical table source

1. Click a source to select it, then click the Edit button to open the General tab of the Logical Table Source dialog box.
2. Complete the Logical Table Source dialog box.

To delete a logical table source

1. Click the source you want to remove and click the Remove button, or press the Delete key.

   A confirmation message appears.

2. Click Yes to remove the source, or No to cancel the removal.

To reorder a logical table source

1. In the Name list, select the source you want to reorder.
2. Use the drag-and-drop feature to reposition the source, or click the Up and Down buttons.

Using the Logical Table Dialog Box—Foreign Keys Tab

Use the Foreign Keys tab to identify or edit the foreign keys for a logical table.

To edit a foreign key

1. In the Keys list, select the key you want to edit.
2. Click Edit.
3. Select or deselect the appropriate columns, and click OK.

To delete a foreign key

- In the Keys list, select the foreign key you want to delete, and then click Delete.

Working with Logical Columns

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. Logical columns in the Business Model and Mapping layer, as well as presentation columns in the Presentation layer, may be reordered.

This section includes the following topics:

- "Creating or Editing a Logical Column"
- "Using the Logical Column Dialog Box—Datatype Tab" on page 157
- "Setting Default Levels of Aggregation for Measure Columns" on page 157
- "Setting Up Dimension-Specific Aggregate Rules" on page 158
- "Using the Logical Column Dialog Box—Levels Tab" on page 160

Creating or Editing 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 or edit a logical column

1. Access the Logical Column dialog box by right-clicking a logical table in the Business Model and Mapping layer, and then selecting New Object > Logical Column from the shortcut menu.
2. Select the General tab.
3. Specify a name for the logical column.

   The name of the column and its associated path appear in the Belongs to Table field.
4. If you want the logical column to be derived from other logical columns:
   - Select the option Use existing logical columns as source.
   - Click the Expression Builder button to open the Expression Builder for derived logical columns.
   - In the Expression Builder, specify the expression from which the logical column should be derived.
   - Click OK to return to the Logical Column dialog box.
5. (Optional) Enter a description of the logical column.

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

Using the Logical Column Dialog Box—Datatype Tab

Use the Datatype tab to view information about the logical column. You can also edit the logical table sources from which the column derives its data, or unmap it from its sources.

To edit the logical table source

1. Select the logical table source you want to edit and double-click it.

   The Logical Table Source dialog box appears.

2. Make your changes, and click OK to return to the Logical Column dialog box.

To unmap a logical column from its source

- Select the logical table source from the list and click the Unmap button.

  The information in the Mapped As list is cleared.

Setting Default Levels of Aggregation for Measure Columns

Use the Aggregation tab of the Logical Column dialog box to set default levels of aggregation for measure columns.

You need to specify aggregation rules for logical columns that are measures (and which are not already mapped to existing logical columns).

TIP: When you specify a default aggregation, that aggregation is always performed on requests for the column. To prevent confusion, you may want to incorporate the aggregation method into the column name. For example, if the column measures sales dollars and the default aggregation is SUM, name the column something like SumOfDollars.

To specify a default aggregation rule for a measure column

1. In the Aggregation tab of the Logical Column dialog box, 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 selects the column in a query.

2. Click OK.

Setting Up Dimension-Specific Aggregate Rules

Use the Aggregation tab of the Logical Column dialog box to set up dimension-specific aggregate rules.

The majority of measures have the same aggregation rule for each dimension. Some measures, however, 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; that is, to specify one aggregation rule that applies to a given dimension and other rules that apply to other dimensions.

You need to have dimensions configured in the Business Model and Mapping layer in order to set up dimension-specific aggregation.

For more information about setting up aggregate navigation, see "Setting Up Aggregate Navigation" on page 199.

To specify dimension-specific aggregation rules for a column

1. In the Aggregation tab of the Logical Column dialog box, select the option Use advanced options.

   The Advanced options section of the dialog box becomes active.

2. Click New to open the Browse window, where you can select a dimension over which you want to perform aggregation.

3. Select a dimension over which you want to perform aggregation and click OK.

   The dimension is added to the list.

4. Click the Expression Builder button to the right of the Features column to open the Expression Builder for Aggregates.

5 You can enter the aggregation to perform over the dimension directly into the Formula area, such as SUM(SalesFacts.PlanRevenue), or you can use the Expression Builder to configure the aggregation.

   For example, if you have a time dimension and you want the calculation to determine the average over time and the sum over the other dimensions, the expression you create in the Expression Builder would be:

   ```
   AVG(<column_name>)
   ```

   where column_name is the name of the logical column.

6 If you have other dimensions for which you want to set up dimension-specific aggregation rules, repeat Steps 2 through 5.

7 Specify the aggregation rules to use for all the other dimensions in the entry labeled Other.

8 Use the Up and Down buttons to specify the order in which the dimension-specific rules are performed. When calculating the measure, the aggregation rules are applied in the order (top to bottom) configured in the dialog box.

9 Click OK to save your modifications.

NOTE: If you use dimension-specific aggregation rules, make sure the users understand the definition of the column. In the example shown in Step 5, the user is not able to determine the sum over time. You would have to create another column to answer that question.

To reorder an aggregation rule

1 In the dimension or formula, select the rule you want to reorder.

2 Use the drag-and-drop feature to reposition the rule, or click the Up and Down buttons.

Using the Logical Column Dialog Box—Levels Tab

Attributes can be associated with a level by selecting the dimensional level on this tab. Use the Levels tab to associate a level-based measure (or any attribute) with multiple dimensions to address your organization's analytical requirements. This allows you to analyze, for example, regional share of product sales by both category and country.

Dimensions are displayed in the Dimensions list. If this attribute is associated with a level, the level is displayed in the Levels list.

The fastest 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. You can also make the associations from the Levels tab. For more information about level-based measures, see "Level-Based Measure Calculations" on page 169.

To associate this measure with a level in a dimension

1. Click in the white space in the Levels list next to the dimension you want to select a level from.

   A drop-down list appears.

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.

Working with 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 logical table to physical table mappings. Complex mappings, including formulas, are also configured in the logical table sources.

Logical tables can and routinely do 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 the 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 by selecting the source folder for a table and selecting New Logical Table Source from the right-click menu.

For examples of how to set up aggregate navigation, see “Setting Up Aggregate Navigation” on page 199.

This section includes the following topics:

- “Creating or Editing a Logical Table Source”
- “Defining Physical to Logical Table Mapping” on page 162
- “Defining Content of Sources” on page 163

Creating or Editing 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 define general properties for the logical table source

1. Access the Logical Table Source dialog box by right-clicking a logical table in the Business Model and Mapping layer, and then selecting New Object > Logical Table Source.
2. Specify a name for the logical table source.
3. Use the Map to These Tables area to specify the physical tables for the source.
4. Click Add to open the Browse dialog box, where you can select a table.

5. To add additional source tables, repeat the previous step for each table.

   When there are two or more tables in a logical table source, all of the participating tables must have joins defined between them. You can view the joins by selecting a table in the Tables area and clicking the View Details button to open the Physical Join dialog box, but you need to define the joins either in the Joins Manager or in the Physical Table Diagram.

6. Use the Joins area to open the Physical Join dialog box, where you can view the join conditions for the tables displayed in the Tables area.

7. (Optional) Enter a description of the table source.

8. To map logical columns to physical columns and specify any transformations that should occur between the Physical layer and the Business Model and Mapping layer, click the Column Mapping tab.

To remove a table as a source

1. In the Tables area, highlight the table you want to remove as a source and click Remove.

2. Repeat this process to remove other tables.

Defining Physical to Logical Table Mapping

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 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. From the Physical Table list on the right side of the Column Mapping tab, select the physical table that contains the column you want to map.

2. Select the physical column in the Expression list corresponding to each logical column.

   When you click on a cell in the Physical Table column, this brings up a drop-down list from which you can see a list of tables currently included in this logical table source.

3. To open the Expression Builder, click the button to the left of the Expression list.

   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.

4. Click on a cell in the Expressions list to see a drop-down list.
5. Click on the arrow to see a list of columns in the physical table you have just specified.
6. Repeat this process to map other columns.

To remove a column mapping

1. Click the Delete button next to the column mapping you want to remove.
2. Repeat this process to remove other column mappings.

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 200.

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 want to define content for dimension levels, do the following:
   a. Select Level from the Aggregation content drop-down list.
   b. In the Level pane, select the appropriate level for the dimension.

      You do not need to specify a level for each dimension if the logical table source is at the lowest level for all dimensions and no dimensions have levels specified. Otherwise, dimensions with no level specified will be interpreted as being at the Grand Total level. Therefore, you should specify a level for each dimension, unless you are specifying 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 the source physical table.
   c. In the Column pane, select the logical columns that map to the keys of the source physical table.
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 enter 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 204.
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 Enter the Where clause, and click OK.

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

Working with Dimensions

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 could contain the attributes day, week, month, quarter, and year. Exactly what attributes a dimension contains depends on the way the business is analyzed.

Dimensions are organized into hierarchies. These hierarchies express the organizational rules set up by your business. Hierarchies provide the metadata needed for the Siebel Analytics Server to drill down within and across dimensions to get less summarized views of the data.

All of the objects within a dimension are used for metadata purposes only. End users do not see the dimensions themselves. The metadata from the dimensional hierarchies is used to perform aggregate navigation, to set up dimension-specific aggregation rules (see “Dimension-Specific Aggregation Rules” on page 210), and to configure level-based measure calculations (see “Level-Based Measure Calculations” on page 169), and to determine what attributes are shown when Siebel Analytics Web users drill down in their data requests.

These hierarchy definitions must 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.

Each dimension can have one or more hierarchies. Within each hierarchy is one or more levels, and each level has one or more attributes associated with it. Each dimension can be associated with attributes from just one logical dimension table (plus level-based measures from the logical fact tables).

NOTE: If the logical table with which a dimension is associated has a foreign key defined, then the key column of the logical table must be included at the lowest level of the dimension.

Creating a Dimension

Use the General tab of the Dimension dialog box to create a dimension.

To create a dimension

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

Working with Levels

A dimensional hierarchy contains two or more levels. The recommended procedure for creating levels is to begin at the top of the hierarchy, creating a grand total level and then creating new child levels, working down the hierarchy to the lowest level. Each dimension can have just one level that is specified as the Grand Total level.

This section includes the following topics:

- "Level Attributes"
- "Level Keys" on page 168
- "Grand Total Levels" on page 168
- "Level-Based Measure Calculations" on page 169
- "Creating or Editing a Level" on page 172
- "Using the Level Dialog Box—Keys Tab" on page 173
- "Using the Level Dialog Box—Preferred Drill Path Tab" on page 173

Level Attributes

Once all levels within a dimension have been created, you need to drag and drop one or more columns from the logical table to each level except the Grand Total level. Logical columns associated with a level are called level attributes. All levels, except the level specified as the Grand Total level, need to have at least one level attribute. However, it is not necessary to associate all of the columns from a table explicitly with levels. All columns not associated with a level will be assumed to be associated with the lowest level of the dimension corresponding to that dimension table. All logical columns from the same logical dimension table have to be associated with the same dimension.

Attributes can also be associated with a level by selecting the dimensional level on the Level tab of the Logical Column dialog box.

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*. Specify the level key in the Level Key dialog box. 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 user double-clicks to drill down. This may or may not be the primary key of the level. To set the level key to display, check the 'Use for drill down' check box on the Level Key dialog box.

Be careful with 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, Year, as shown. To add Year, click the Add button in this dialog and select the logical column from the dialog that is presented.

Grand Total Levels

A *grand total level* is a special level representing the grand total for a dimension. To specify a level as a grand total level, check the option 'Grand Total Level' in the General tab of the Level dialog box. Grand total levels do not require level keys.

You can associate measures with a grand total level. For example, consider a product dimensional hierarchy with levels TotalProducts (grand total level), Brands, and Products. Consider also that there is 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.

Level-Based Measure Calculations

A *level-based measure* is a column whose values are always calculated based on a particular level. 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 each of these—for example, CountryRevenue, RegionRevenue, and CityRevenue. The measure AllProductRevenue discussed in the previous example 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 is to create the metadata that defines a one-to-many relationship between Country and Region and a one-to-many relationship between Region and City; that is, 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.

You then need to create three logical columns—CountryRevenue, RegionRevenue, and CityRevenue. Each of these columns uses the logical column Revenue as its source (in the General tab of the Logical Column dialog). The Revenue column has a default aggregation rule of SUM (in the Aggregation tab of the Logical Column dialog) and has sources in the underlying databases.

You then drag the CountryRevenue, RegionRevenue, and CityRevenue columns into the Country, Region, and City levels, respectively. Each query that requests one of these columns will return the revenue based on the share specified by the level. Figure 13 shows what the business model in the Business Model and Mapping layer would look like to model this example.

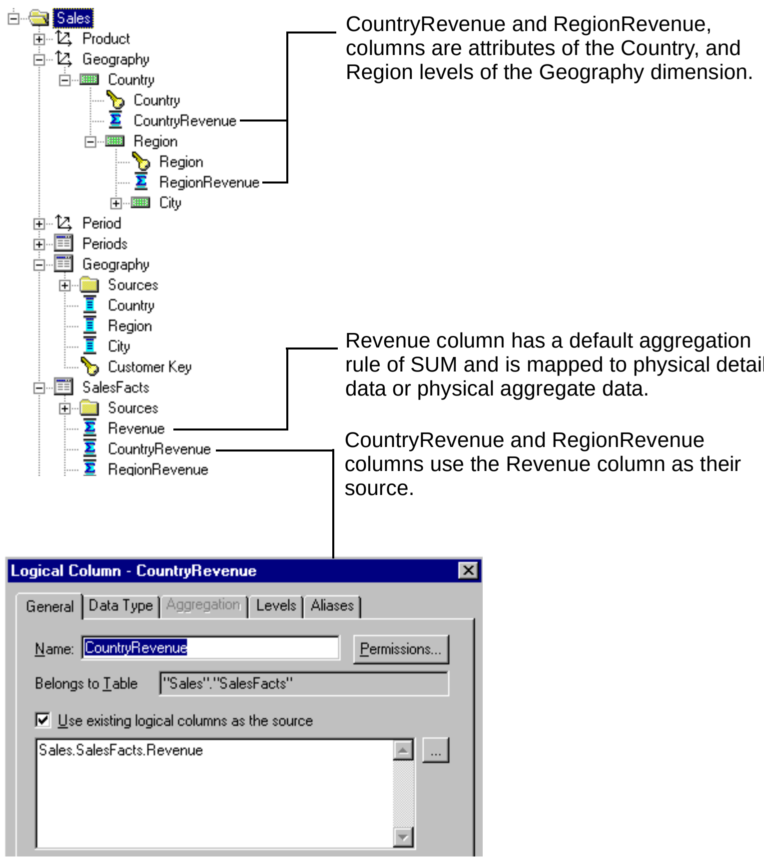


Figure 13. Example Business Model in the Business Model and Mapping Layer

You can also set the level of a measure (or any attribute) by using the Levels tab of the Logical Column dialog box.

Creating or Editing a Level

Use the General tab to create or edit a level in the Business Model and Mapping layer.

To define general properties for the 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.
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 option Supports rollup to parent elements.
    - Is not the grand total level and does not roll up, leave both options unselected.
4. To define child levels, click Add.
5. 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.
6. 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.
7. (Optional) Enter a description of the level.
8. Select and complete the Keys tab. Grand Total levels are not required to have keys, although you may specify one if you wish (for example, a column returning the constant 'All Countries').

Using the Level Dialog Box—Keys Tab

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

To add a key

1. Click New to open the Level Key dialog box for levels or the Logical Key dialog box for tables.
2. Select the columns that comprise the key.
3. Click OK to return to the Level dialog box or Logical Table dialog box.

   The key is added to the Primary Key drop-down list.

To edit a key

1. Select the key you want to edit and either double-click it or click the Edit button to open the Level Key dialog box for levels or the Logical Key dialog box for tables.
2. Make your changes, and then click OK.

Using the Level Dialog Box—Preferred Drill Path Tab

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 hierarchy. This feature is most commonly used to drill from one dimension to another.

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.

To delete a level from the drill path

1. Select the level you want to delete.
2. Click Remove, or press the Delete key.

To reorder a level in the drill path

1 In the Names list, select the level you want to reorder.

2 Use the drag-and-drop feature to reposition the level, or click the Up and Down buttons.

About 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. If the logical joins are the same as in the Physical layer, they will be set up automatically if the logical tables are created by dragging and dropping all of the participating physical tables simultaneously to the Business Model and Mapping layer. However, you should 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. Complex joins are joins over nonforeign key-primary key columns; that is, the relationship between the tables is arbitrary. 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. A complex join in the Business Model and Mapping layer acts as a placeholder when the business model has no foreign keys but requires a physical join to be created with expression and column information. A complex join in the Physical layer does not require a matching join in the Business Model and Mapping layer.

For information about how to create logical joins, see "Defining Logical Joins with the Joins Manager" on page 176 and "Defining Logical Joins with the Business Model Diagram" on page 178.

You can create primary keys for tables using the Key tab of the Logical Table dialog box. For more information, see "Using the Logical Table Dialog Box—Key Tab" on page 153.

You can create primary keys for levels using the Key tab of the Level dialog box. For more information, see "Using the Level Dialog Box—Keys Tab" on page 173.

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 177

Creating a Logical Foreign Key

Use this procedure to create a logical foreign key.

To create a logical foreign key

1. In the Administration Tool tool bar, 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. Enter 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 180.

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

Use this procedure to create a logical complex join.

To create a logical complex join

1 In the Administration Tool tool bar, select Manage > Joins.

The Joins Manager dialog box appears.

2 Select Action > New > Logical Complex Join.

The Logical Join dialog box appears.

3 Enter 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 180.

   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 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 save your work.

Defining Logical Joins with the Business Model Diagram

The Business Model Diagram shows logical tables and any defined joins between them. You can use the Business Model Diagram to define logical foreign keys and complex joins.

To display the Business Model Diagram

1 In the Administration Tool, right-click a business model, and then select Business Model Diagram.

2 Click one of the following buttons on the Administration Tool tool bar:

   - New foreign key
   - New complex join

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 180.

   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.

   The expression displays in the Expression pane.

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.

Showing Physical Tables That Are Sources of Logical Objects

You can execute the Physical Diagram command when you have selected a business model, logical table, or logical table source. This displays a diagram showing all of the physical tables that map to the selected logical object.

Administration Tool Utilities and Expression Builder 8

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 184
- Synchronize Aliases on page 184
- Replace Wizard on page 185
- Copy Business Model with Presentation Catalog on page 185
- Siebel Analytics Event Tables on page 186
- Execute UDML on page 186
- Query Statistics on page 186
- Externalize Strings on page 187
- Rename Wizard on page 187
- Update Physical Layer Wizard on page 188
- Repository Documentation on page 191

Time Series Wizard

The Time Series Wizard automates the process of creating measures and calculations to support historical time comparison analyses.

To start the Time Series Wizard

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

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 name changes in the original table, this utility adjusts the alias accordingly.

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 the right-click command ‘Duplicate Business Model with Catalog Folder’ 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 for use in Siebel Analytics Server caching activities. Using 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 stale cache entries that reference those physical tables. For more information about event tables, see "Cache Event Processing with an Event Polling Table" on page 255.

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 Database Markup Language).

To start the Execute UDML utility

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

Query Statistics

This utility allows you to control the collection of detailed query statistics for users and groups. Detailed query statistics provide a comprehensive analysis of the query load (both logical queries and back-end database queries) against the Siebel Analytics Server.

To start the Query Statistics utility

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

Externalize Strings

This utility is primarily for use by translators to translate internal text strings displayed in the Administration Tool. It allows Siebel Analytics Web users using a language other than English (US) to see the names of catalogs, tables and columns, and their descriptions, in their own language.

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.

   Clicking on the Presentation Catalogs in the left pane shows, in the right pane, the translated values and the original strings (names), which will be placed in session variables for use by Siebel Analytics Web.
3. Click Save to save the strings in the selected format.
4. Click the Close button to end the utility.

Rename Wizard

The Rename Wizard allows you to rename items in the workspace.

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 "Working with Connection Pools" on page 121.

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.

3. If you have Siebel Analytics applications, you can mark the Siebel tables in the repository for updating by clicking the selection box next to each database to add a check mark.

   NOTE: Select this option only if the option to import Siebel tables was selected when the objects were imported into the Physical layer.

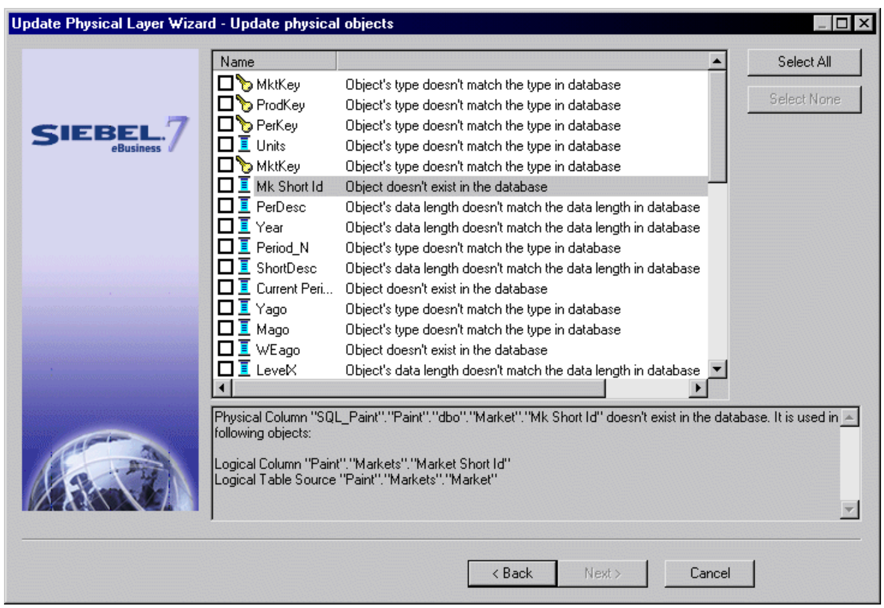
4 When you are done selecting databases to update, click Next.

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 doesn’t exist in the database

5 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.



6 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.

7 Review the information about each update.

8 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.

9 Click Finish.

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

10 On the Administration Tool toolbar, click File > Save to save the updated objects in the Physical layer.

Repository Documentation

This 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 is output in Microsoft Excel format.

To run the Repository Documentation utility

1. From the Administration Tool toolbar, select Tools > Utilities > Repository Documentation, and then click Execute.

   The Save As dialog box appears.

2. Select the location where you want to save the file.
3. Enter a name for the file, and then click OK.

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 373. For information about the SQL functions supported by the Siebel Analytics Server, see "SQL Reference" on page 386.

This section includes the following topics:

■ "About the Expression Builder Dialog Boxes"

■ "Expression Builder Toolbar" on page 194

■ "Folders in the Selection Pane" on page 195

■ "Example of Setting Up an Expression" on page 196

■ "Navigating Within the Expression Builder" on page 197

■ "Building an Expression" on page 198

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 14 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.

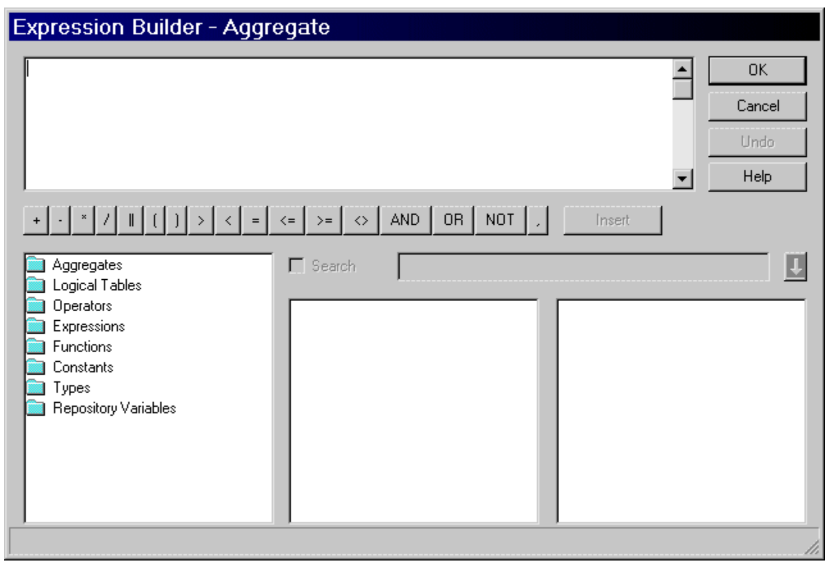


Figure 14. Example Expression Builder

Expression Builder Toolbar

The toolbar is located in the middle portion of the expression builder. Table 14 describes each icon and its function in an expression.

Table 14. Expression Builder Toolbar

<table>
<tr><th>Operator</th><th>Description</th></tr>
<tr><td>+</td><td>Plus sign for addition.</td></tr>
<tr><td>-</td><td>Minus sign for subtraction.</td></tr>
<tr><td>*</td><td>Multiply sign for multiplication.</td></tr>
<tr><td>/</td><td>Divide by sign for division.</td></tr>
<tr><td>||</td><td>Character string concatenation.</td></tr>
<tr><td>(</td><td>Open parenthesis.</td></tr>
<tr><td>)</td><td>Close parenthesis.</td></tr>
<tr><td>></td><td>Greater than sign, indicating values higher than the comparison.</td></tr>
<tr><td><</td><td>Less than sign, indicating values lower than the comparison.</td></tr>
<tr><td>=</td><td>Equal sign, indicating the same value.</td></tr>
<tr><td>< =</td><td>Less than or equal to sign, indicating values the same or lower than the comparison.</td></tr>
<tr><td>> =</td><td>Greater than or equal to sign, indicating values the same or higher than the comparison.</td></tr>
<tr><td>< ></td><td>Not equal to, indicating values higher or lower, but not the same.</td></tr>
<tr><td>AND</td><td>AND connective, indicating intersection with one or more conditions to form a compound condition.</td></tr>
<tr><td>OR</td><td>OR connective, indicating the union with one or more conditions to form a compound condition.</td></tr>
<tr><td>NOT</td><td>NOT connective, indicating a condition is not met.</td></tr>
<tr><td>,</td><td>Comma, used to separate elements in a list.</td></tr>
</table>

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 15 shows the expression builder for a derived logical column, with a blank expression.

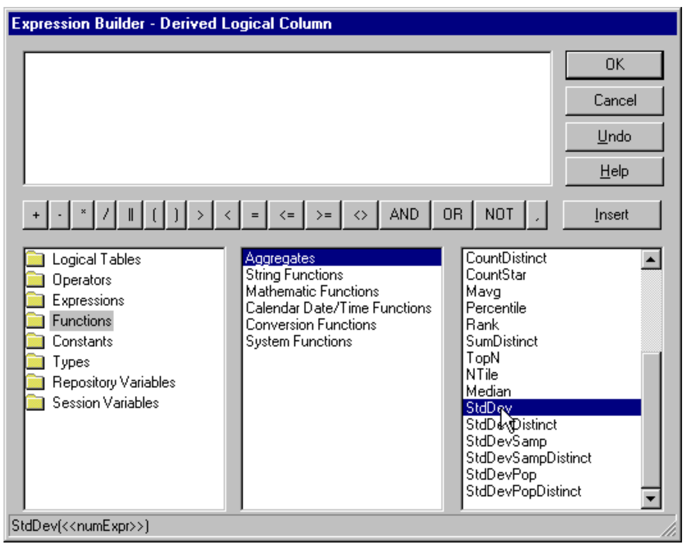


Figure 15. Expression Builder for Derived Logical Columns

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 16 shows where the expression appears in the window.

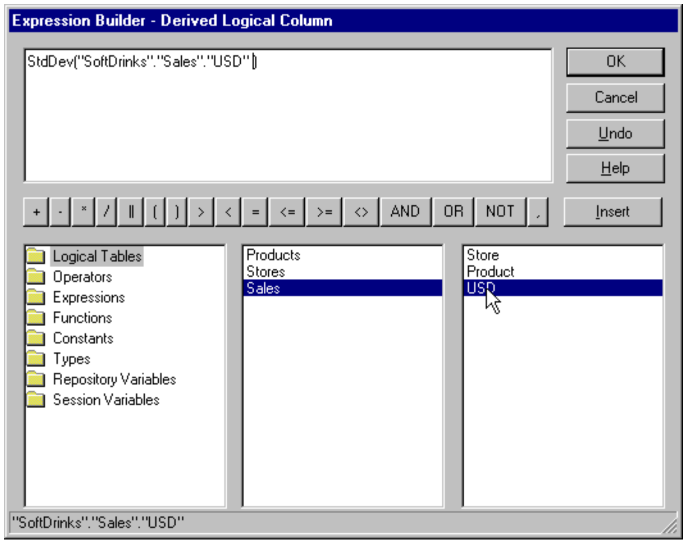


Figure 16. Example Logical Column Function in the Editing Pane

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 9

This section includes a description of how you can use aggregate navigation and provides setup instructions.

About Aggregate Navigation

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.

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 provides the answers the fastest.

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 200
- "Create Dimension Sources for Each Level of Aggregated Fact Data" on page 202
- "Specify Fragmentation Content" on page 204

- "Dimension-Specific Aggregation Rules" on page 210
- "Aggregate Table Fragments" on page 211

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. If the granularity of the data is at the sales transaction level, then this aggregate table contains highly summarized data.

Such an aggregate table could be used to answer the following logical SQL request:

```
select product_name, store_name, month, sum(revenue)
from product, store, time, facts
```

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 163.

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.

For example, if you have an aggregate source that includes all soda brands, but you are only interested in the data from Fizzy Brands, the following is a valid WHERE clause filter, where Brand is a column of the physical table Product:

```
Product.Brand = 'Fizzy Brands'
```

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 the source that references an aggregate table, you define a content specification, as described in "Specify Content of Source" on page 200. The content specification (in the Advanced tab of the Logical Table Source window) in the example in "Specify Content of Source" on page 200 specifies an aggregate fact table at the Product.Product_Name, Store.Store_Name, and Time.Month levels. In addition to the source for the aggregate fact table, you should create corresponding logical dimension table sources at the same levels of aggregation.

Using this example, you need to create 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:

  ```
  GROUPBYLEVEL (ProductDimension.ProductLevel)
  ```

  or

  ```
  GROUPBYCOLUMN (Product.Product_Name)
  ```

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

  ```
  GROUPBYLEVEL (StoreDimension.StoreLevel)
  ```

  or

  ```
  GROUPBYCOLUMN (Store.Store_Name)
  ```

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

  ```
  GROUPBYLEVEL (TimeDimension.MonthLevel)
  ```

  or

  ```
  GROUPBYCOLUMN (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.

You do not have to use GROUPBYCOLUMN or GROUPBYLEVEL exclusively. You can mix the use of both expressions.

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, as described in "Dimension-Specific Aggregation Rules" on page 210.

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, enter 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. To use the Expression Builder to configure the content specification, click the button to the right of the content pane.
6. Set any necessary WHERE clause constraints.
7. Repeat this process for each source corresponding to the logical tables in the aggregate table's content specification.

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 13, "Using Variables in a 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 206.

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. Once the order has shipped, however, the only change that can be made to the order is to enter 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.

Dimension-Specific Aggregation Rules

Explicitly defined dimensions in the Business Model and Mapping layer provide a mechanism to set up level-based calculations, which always calculate values at a given level. For information on level-based calculations, see "Level-Based Measure Calculations" on page 169.

The majority of measures have the same aggregation rule for each dimension they aggregate over. Some measures, however, can have different aggregation rules for different dimensions. For example, bank balances might be summed over individual accounts but averaged over time. The Siebel Analytics Server allows you to configure dimension-specific aggregation rules; that is, to specify one aggregation rule that applies to a given dimension and other rules that apply to other dimensions.

You need to define explicit dimensions to configure dimension-specific aggregation. For details on configuring a measure to use dimension-specific aggregation, see "Setting Up Dimension-Specific Aggregate Rules" on page 158.

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 17, 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.

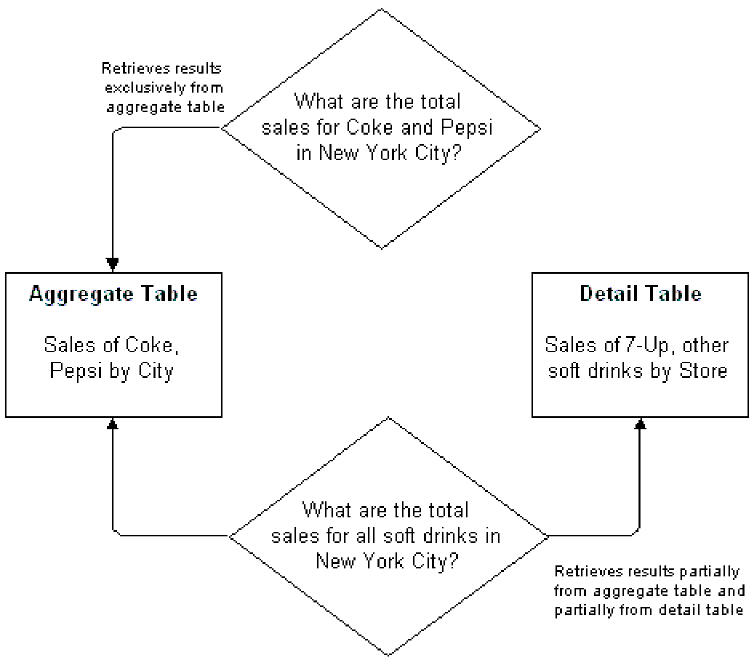


Figure 17. Aggregating Information

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:

```
GROUPBYLEVEL(SoftDrinks.GeographyDim.CityLevel,
SoftDrinks.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 204.

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 enter the SQL statement in the pane to the right.

The SQL needs to define a virtual table that completes the domain at the level of the other aggregate tables. In this case, there is one existing aggregate table, and it contains data for Coke and Pepsi by city. Therefore, the SQL statement has to return all of the data at the city level, except for the Coke and Pepsi data.

Figure 18 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.

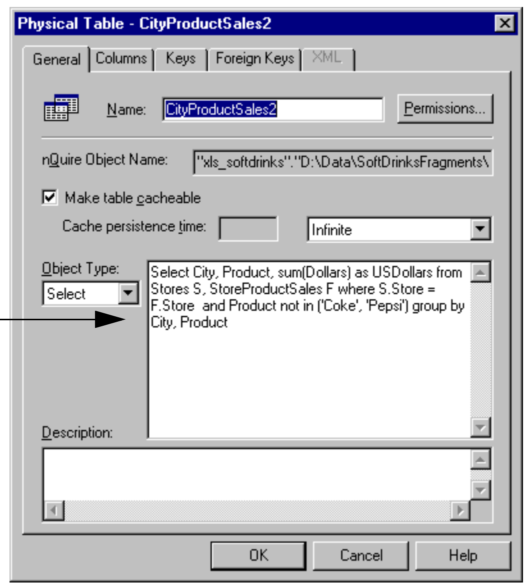


Figure 18. Example Physical Table Definitions

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:

```
GROUPBYLEVEL(SoftDrinks.GeographyDim.CityLevel,
SoftDrinks.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 19.

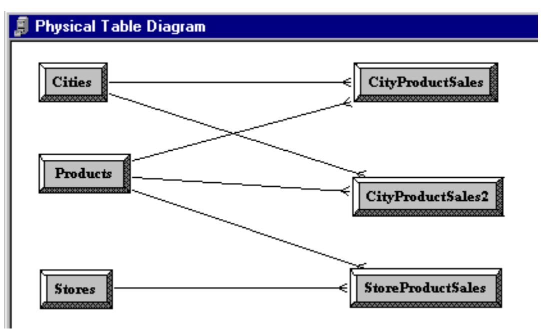


Figure 19. Example Physical Joins

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 10

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:

  ```
  nqscomgateway.exe &
  ```

- For the standard shell:

  ```
  nohup nqscomgateway.exe >/dev/null 2>&1 &
  ```

- For the C shell:

  ```
  nohup nqscomgateway.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 Enter 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 NQSConfig.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 Siebel Analytics Server 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

- Enter 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:

data_source_name	The name of a nonclustered ODBC data source used to connect to the server to perform the shut down operation. The data source needs to connect to the server as 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.

user ID	The user to connect to the Siebel Analytics Server to perform the shut down operation.
password	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 Siebel Analytics Server 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 306.

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 269.

You can also run the Siebel Analytics Server 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 NQSConfig.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 15.

Table 15. Logging Levels

Logging Level	Information That Is Logged
Level 0	No logging.
Level 1	Logs the SQL statement issued from the client application. Logs elapsed times for query compilation, query execution, query cache processing, and backend database processing. Logs the query status (success, failure, termination, or timeout). Logs the user ID, session ID, and request ID for each query.
Level 2	Logs everything logged in Level 1. 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.
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.

Table 15. Logging Levels

Logging Level	Information That Is Logged
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:

user_ID	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.
log_input_filename	The name of an existing log file. If not specified, the active log file is read.
output_result_filename	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.
session_ID	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.
request_ID	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 236.

Interpreting the Log Records

Once 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, which capture valuable information about users, performance, and so on, 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 usage tracking is enabled, statistics for every query are written to a usage tracking log file. User-defined intervals control the frequency at which the data is written to disk and control new usage tracking log files created. (See *Siebel Analytics Installation and Configuration Guide* for details.)

This section describes the usage tracking system and includes information about selecting a location for usage tracking log files, file naming conventions, format of the log file, and performance considerations.

Because there is no limit to the number or size of the usage tracking log files, it is important to consider how these files will be handled before enabling usage tracking.

Selecting an Output Location

The parameter STORAGE_DIRECTORY in the Usage Tracking section of the NQSConfig.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 16. For more information about the contents of each column, see "Description of the Usage Tracking Data" on page 443.

Table 16. 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

Table 16. Usage Tracking Output File Format

Column Number	Column Name	Data Type	Max Data Size	Nullable
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
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 "Sample Scripts and Load Procedures for Usage Tracking Data" on page 435.

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 theNQSConfig.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 Siebel Analytics Server and all current query requests for each user. The Siebel Analytics Server administrator can disconnect any users and kill any query requests with the Session Manager.

Using the Session Manager

The Session Manager contains two windows:

- The top window, the Session window, shows users currently logged into the Siebel Analytics Server.
- The bottom window, the Request window, shows active query requests for the user selected in the Session window.

Table 17 and Table 18 describe the columns in the Session Manager windows.

Table 17. 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 18. 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 *Siebel Analytics Server Administration Tool Online Help* for the meanings of the different statuses.
Description	Not used.

To open the Session Manager

- In the Administration Tool, open a repository in online mode, and then select Manage > Sessions.

To disconnect a user from a session

1. Select the user in the Session Manager top window.
2. 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.

To kill an active query

1. Select the user session that initiated the query in the top window of the Session Manager.

   Once the user is highlighted, any active query requests from that user are displayed in the bottom window.

2. Select the request you want to kill.
3. 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.

To control the update speed

- Select Normal, High or Low from the Update Speed drop-down list. Select Pause to keep the display from being refreshed.

  How often the Session Manager display refreshes is a function of the amount of activity on the system.

- To refresh the display at any time, click the Refresh button.

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 199.

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 11, "Query Caching."

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 225.

Query Caching 11

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 Query Cache

The *query cache* in the Siebel Analytics Server is a facility that stores the results from queries. The results in the cache can be used as a source from which to aggregate, that is, the cache can be a source for queries that are at a higher level of aggregation.

The Siebel Analytics Server stores metadata about each stored query result. The cache metadata is used 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*.

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 15, “Security.”

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.

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. Figure 20 shows the basic architecture of the query cache and illustrates how query caching can remove many of the most costly steps.

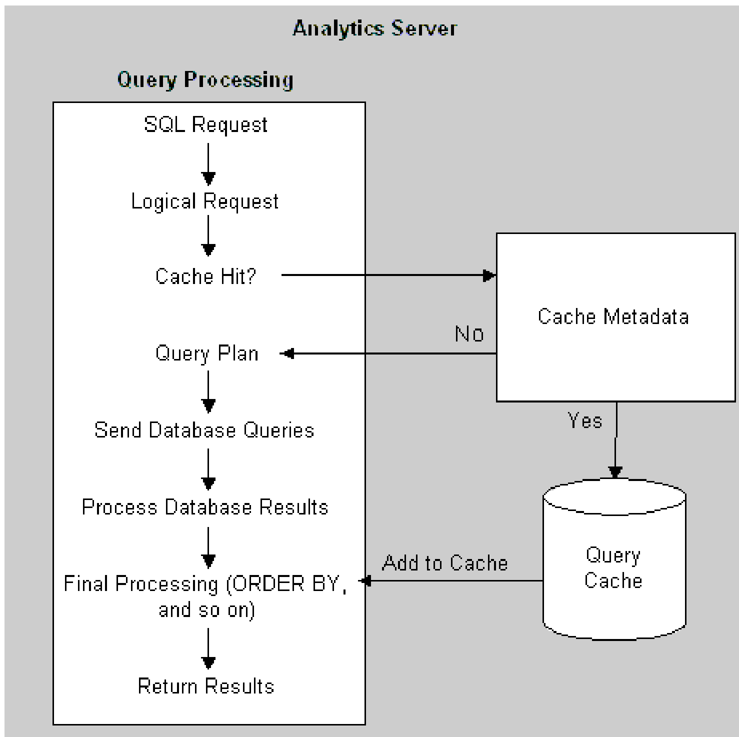


Figure 20. Example Physical Table Definitions

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 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 251.

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*.

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 249, 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 reenable 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 Siebel Analytics Server 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.

Once 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 Siebel Analytics Server Administration Tool and clear the option Make table cachable in the General tab of the Physical Table dialog box. Once 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 256.

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 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.

- **No additional aggregation.** For example, if a query with the column *qtysold* is cached, a request for RANK(qtysold) results in a cache miss.

- **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.

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 19 on page 256. 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 262). 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. Once 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 19; 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 19. Sample CREATE TABLE statements to create an event polling table are shown in "Sample Event Polling Table CREATE TABLE Statements" on page 259.

Table 19. 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 DEFAULT CURRENT_TIMESTAMP for Oracle 8i. Values cannot be NULL.

Table 19. Event Polling Table Column Names

Event Table Column Name	Datatype	Description
DatabaseName	CHAR or VARCHAR	The name of the database where the physical table that was updated resides. This is the name of the database as it is defined in the Physical layer of the Administration Tool. For example, if the physical database name is 11308Production, and the database name that represents it in the Siebel Analytics Server Administration Tool is SQL_Production, the polled rows in the event table has to contain SQL_Production as the database name. Populate the DatabaseName column only if the event table does not reside in the same database as the physical tables that were updated. Otherwise, set it to the NULL value.
CatalogName	CHAR or VARCHAR	The name of the catalog where the physical table that was updated resides. Populate the CatalogName column only if the event table does not reside in the same database as the physical tables that were updated. Otherwise, set it to the NULL value.
SchemaName	CHAR or VARCHAR	The name of the schema where the physical table that was updated resides. Populate the SchemaName column only if the event table does not reside in the same database as the physical tables being updated. Otherwise, set it to the NULL value.
TableName	CHAR or VARCHAR	The name of the physical table that was updated. The name has to match the name defined for the table in the Physical layer of the Administration Tool. Values cannot be NULL.
Reserved	CHAR or VARCHAR	Reserved for future enhancements. The Reserved column is not populated but has to exist.

Table 19. Event Polling Table Column Names

Event Table Column Name	Datatype	Description
DatabaseName	CHAR or VARCHAR	The name of the database where the physical table that was updated resides. This is the name of the database as it is defined in the Physical layer of the Administration Tool. For example, if the physical database name is 11308Production, and the database name that represents it in the Siebel Analytics Server Administration Tool is SQL_Production, the polled rows in the event table has to contain SQL_Production as the database name. Populate the DatabaseName column only if the event table does not reside in the same database as the physical tables that were updated. Otherwise, set it to the NULL value.
CatalogName	CHAR or VARCHAR	The name of the catalog where the physical table that was updated resides. Populate the CatalogName column only if the event table does not reside in the same database as the physical tables that were updated. Otherwise, set it to the NULL value.
SchemaName	CHAR or VARCHAR	The name of the schema where the physical table that was updated resides. Populate the SchemaName column only if the event table does not reside in the same database as the physical tables being updated. Otherwise, set it to the NULL value.
TableName	CHAR or VARCHAR	The name of the physical table that was updated. The name has to match the name defined for the table in the Physical layer of the Administration Tool. Values cannot be NULL.
Reserved	CHAR or VARCHAR	Reserved for future enhancements. The Reserved column is not populated but has to exist.

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 Siebel Analytics Server 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.

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(20) null,
  CatalogName varchar(20) null,
  SchemaName  varchar(20) null,
  TableName   varchar(20) not null,
  Other       varchar(10) 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(20) null,
  CatalogName varchar(20) null,
  SchemaName  varchar(20) null,
  TableName   varchar(20) not null,
  Other       varchar(10) null
);
```

You might need to modify these CREATE TABLE statements slightly for different versions of SQL Server and Oracle, or for other databases. Also, 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

Once 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.

Once 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 225.

Making Changes to a Repository

When you modify Siebel Analytics Server repositories, the changes can have implications on 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 266.

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 tool bar, 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 20.

Table 20. 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 *stale* 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 20. 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 21 describes the information that appears in the Global Cache Information window.

Table 21. 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 21. Global Cache Information

Column	Description
Amount of space used on disks containing cache related files	The *total* 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 Siebel Analytics Server 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 Siebel Analytics Server 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 Siebel Analytics Server Administration Tool, see "Using XML as a Data Source" on page 343.

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 requeried 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.

Connectivity and Third-Party Tools 12

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)

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. Enter a name for the data source in the Name field.

5. (Optional) Enter 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, enter 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. Enter 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), enter 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, enter 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 enter 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 enter 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 enter 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 Siebel Analytics Server 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 or Editing a Connection Pool" on page 121. 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.

For the metadata import procedure, see "Importing the Physical Schemas" on page 83.

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 a Repository 13

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 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*.

To access the Variable Manager dialog box

- Select Manage > Variables from the Administration Tool tool bar.

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 279

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

Once 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 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.

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 15, "Security."

This section includes the following topics:

- "Using System Session Variables"
- "Using Nonsystem Session Variables" on page 283

For information about creating a new session variable, see "Creating New Variables" on page 284.

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 22 describes the available system session variables.

Table 22. System Session Variables

Variable	Description
USER	Holds the value the user enters as his or her logon name.
GROUP	Contains the groups that the user belongs to. These are used by both the Siebel Analytics Server and Siebel Analytics Web. 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. 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: `Sales US;Sales UK;QA and Dev;Doc` 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.
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 once 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 22. 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.

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. Select Action > New from the Variable Manager menu and then select the type of variable you want to create from the submenu.

   The Variable dialog box appears. The title bar reflects the type of variable you selected:

   - Static Repository Variable
   - Dynamic Repository Variable
   - Session Variable

2. Enter 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.

3. (Optional) Construct the value for the variable using the Expression Builder:

   a. Click the Expression Builder button to the right of the Default initializer window to open the Expression Builder for Variables.

   b. See "SQL Logical Operators" on page 383 for information on creating the value.

      The expression displays in the Default initializer window.

4 Alternatively, you can enter 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 (‘).

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, or click the New button to open the General tab of the Initialization Block dialog box, where you can create a new initialization block.

If you are constructing a new initialization block, see “Creating and Editing Initialization Blocks” on page 290 for more information. When you are finished, click OK to return to the Variable dialog box.

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 225.

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 225.

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 enter 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 290).

For the initialization string, you enter 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 enter 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

Use the following procedure to create a new initialization block or to edit properties of an existing initialization block.

To create or edit an initialization block

1. In the Administration Tool, select Manage > Variables from the menu.

   The Variables Manager appears.

2. Select Action > New > Initialization Block.

3. Enter 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 288.

   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. Enter the SQL initialization string needed to populate the variables.

6. Specify a refresh interval in the Refresh Time area:

   a. Select a start date and time.

      You can specify the day, the month, the day of the month, the year, and the hours, minutes and seconds.

   b. Enter the periodic duration of the refresh interval in either days, hours, or minutes.

      NOTE: The Refresh Time area only applies to repository variables. It will be 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 an SQL tool such as the Siebel Analytics Client utility. Use the same DSN (or one set up identically) as the one named 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.

   a. To select the connection pool associated with the database where the target information is located, click the Browse button to the right of the Connection Pool field to open the Browse dialog box.

   b. Select the connection pool to use and click OK to return to the Initialization Block dialog box.

      The name of the connection pool appears in the Connection Pool field.

      NOTE: If you do not select the connection pool before typing the initialization string to use, you will receive a message prompting you to select the connection pool to use. You can click the Test button to test the connection.

      ---

9 If you selected Database in the Data Source Connection area, enter 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) You can test the test the initialization block by clicking Test in the Data Source Connection area.

A message box informs you whether the test was successful.

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.

   The Variable Manager dialog box appears.

2. 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 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.

Clustering Siebel Analytics Servers 14

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.

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 Analytics 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 NQSConfig.INI file, the REPOSITORY_PUBLISHING_DIRECTORY parameter specifies the location of the repository publishing directory.

Cluster Manager

The Cluster Manager is available in the Analytics 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 NQSConfig.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 NQSConfig.INI file (when CLUSTER_PARTICIPANT is set to YES in the NQSConfig.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 Analytics Administration Tool connects for online repository changes. Online repository changes are published to the location specified by the REPOSITORY_PUBLISHING_DIRECTORY parameter in the NQSConfig.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 269.

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 Analytics 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.

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 enter 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.

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 NQSConfig.INI file. When set to YES, if the publishing directory is not available at startup or if an error is encountered while the server is reading any of the files in the directory, an error message is logged in the NQServer.log file and the server shuts down. When set to NO, the server joins the cluster and a warning message is logged in the NQServer.log file, but any online repository updates are not reflected in the server’s Repository directory.

6 The primary and secondary Cluster Controllers begins to exchange heartbeat messages with each server that is to participate in the cluster. The connection status is logged in the NQServer.log files of all servers in the SERVERS list. Messages are also logged in the Cluster Controller's NQCluster.log file. Any servers with connection problems are not be allowed to join the cluster. If the server defined as the MASTER_SERVER for online repository is not available, no repository editing in online mode will be possible.

7 As each Analytics Server in the cluster is started, it examines the repository publishing directory for any updated repositories. This is done by comparing the date and timestamps. (The 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 Analytics 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 Siebel Analytics Administration Tool, open a repository in online mode.
2. Select Manage > Clusters.

   The Cluster Manager window appears.

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 Analytics Administration Tool is connected. Table 23 describes categories that may appear.

Table 23. 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: ■ Controller - The object is a Cluster Controller. ■ Server - The object is a Siebel Analytics Server.
Role	The role of the object in the cluster: ■ 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 23. Status View Columns

Column	Description
Status	The status of the object in the cluster: ■ 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	The timestamp showing when the Cluster Controller or Analytics Server was last started. This field will be blank if the Cluster Controller or clustered server is offline.
Last Reported Time	The time the Cluster Controller or Analytics Server communicated with the Controlling Cluster Controller. If the server or controller is offline, this field may be blank.
Sessions	This field is available when either Servers or an individual server is selected in the Explorer pane. It shows the number of sessions currently logged on to a clustered server.

Cache Information

The Cache view is available in the Cluster Manager window if caching is enabled.

The categories of information and their display sequence are controlled by your Options settings. Table 24 describes categories that may appear.

Table 24. 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 25.

- 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 25 describes the information that appears in the Session window.

Table 25. 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 26 describes the information that appears in the Request window.

Table 26. Request Window Columns (Bottom Window)

Column	Description
Session ID	Unique internal identifier that Analytics Server assigns each session when the session is initiated.
Request ID	Unique internal identifier that Analytics Server assigns each query when the query is initiated.
Start Time	Time of the initial query request.
Last Active Time	Timestamp of the last activity on the session or the query.
Status	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. ■ 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.

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*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 15

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.

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 319
- "Importing Users and Groups from LDAP" on page 325

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 337 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 318.

This section includes the following topics:

- "Adding a New User to a Repository" on page 317
- "Siebel Analytics Server Administrator Account" on page 318

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 Siebel Analytics Server Administration Tool.
2. Display the security manager by selecting Manage > Security.
3. Select Action > New > User to open the User dialog box.
4. Enter 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 226 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.
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, enter 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 or Editing a Connection Pool" on page 121.

---

11 Set up any query permissions for the user. For information, see "Query Execution Privileges" on page 339.

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.

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 281.

This section includes the following topics:

- “Predefined Administrators Group”
- “Defined Groups” on page 320
- “Group Inheritance” on page 320
- “Adding a New Group” on page 323
- “Viewing Member Hierarchies” on page 324

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 323.

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 21.

Because privileges granted directly to the user take precedence over those granted through groups, User1 has the privilege to read TableA.

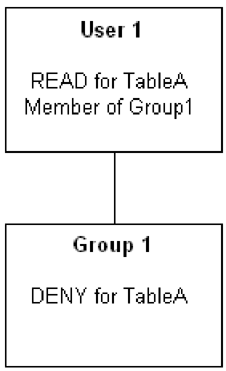


Figure 21. User Privileges and Group Privileges

Example 2

Consider the situation shown in Figure 22.

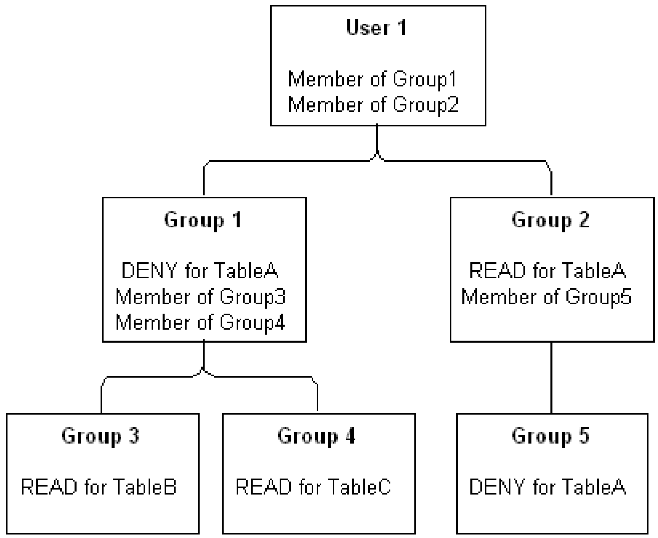


Figure 22. Privileges Example

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 Siebel Analytics Server 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. Enter 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 "Query Execution Privileges" on page 339.

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. Once 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 330.

This section includes the following topics:

- "Configuring an LDAP Server"
- "Importing Users from LDAP" on page 327
- "Synchronizing Users and Groups with LDAP" on page 328

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 Siebel Analytics Server Administration Tool in offline or online mode.
2. Display the security window by selecting Manage > Security.

3 Select Action > New > LDAP Server. Alternatively, you can select LDAP Servers in the tree in the left pane, right-click on white space in the right pane, and select New LDAP Server from the context-sensitive (right-click) menu.

4 Enter the information requested in the LDAP Server Initialization Block dialog box.

- **Host name.** The name of your LDAP server.
- **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.
- **Test Connection.** Use this button to verify your parameters by testing the connection to the LDAP server.

5 Click the Advanced tab, and enter 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 Siebel Analytics Server 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 context-sensitive (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 context-sensitive 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 330
- "External Table Authentication" on page 332
- "Database Authentication" on page 335
- "Siebel Analytics Server Internal Authentication" on page 336

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 enter 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 is not relevant for Siebel Analytics Web users.

To configure operating system authentication

1. Enable operating system authentication by specifying the PERFORM_OS_AUTHENTICATION = YES security option in the NQSConfig.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 281.

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 Variable Manager" on page 277.

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 281.

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. Once 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 325.

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 332.

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 226 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 done. If external table authentication fails, then database authentication is performed.

See “Database Authentication” on page 335, and “Order of Authentication” on page 337 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 Variable Manager" on page 277.

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. Once 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 281.

NOTE: 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 330.

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 context-sensitive menu.

   The Initialization Block dialog box appears.

5. Enter 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, enter 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.

Database authentication can be used in conjunction with external table authentication. If external table authentication succeeds, then database authentication is not done. If external table authentication fails, then database authentication is performed.

See "External Table Authentication" on page 332 and "Order of Authentication" on page 337 for additional details.

NOTE: 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 337 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. Click 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, enter the new password.

6 In the Confirm Password text box, enter the password again, and then click OK.

Order of Authentication

If the user does not enter 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 "Working with Connection Pools" on page 121.

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.

Query Execution Privileges

The Siebel Analytics Server allows you to exercise a fine degree of control over the information in a repository that a user can access.

Controlling Query Privileges

Controlling users' query privileges allows you to manage the query environment to the extent that you desire. You can put a high degree of query controls on users, no controls at all, or somewhere in between. Some types of activities you may want to limit are:

- Controlling runaway queries.
- Restricting users to certain rows or certain columns.
- Restricting query access to certain time periods.

All restrictions and controls can be made at the user level, at the group level, or a combination of the two. The next section provides details about setting up query execution privileges.

Specifying Query Execution Privileges

The following procedure explains how to specify explicit query privileges for a user or group.

To specify explicit query privileges

1. To specify rights for a user or a group explicitly, click Permissions in either the User or Group dialog box.

   The User/Group Permissions dialog box appears.

   By clicking Add, you can choose any object in the repository and explicitly allow or disallow access to the object. If the box in the Read column has a check mark in it, the user or group is granted read privileges on the object. If the box in the Read column has an X through it, the user is disallowed read privileges on the object. If the box in the Read column is blank, no action is taken on the object; that is, any existing privileges (for example, through a group) on the object apply.

   If you explicitly deny access to an object that has child objects, the child objects are also denied access. 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.

   NOTE: 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 320.

2. To set limits explicitly on the time periods when the user or group has access, the number of rows returned, or the maximum database processing time for each of the underlying databases, open the Query Limits tab of the User/Group Permissions dialog box.

3 To specify a maximum number of rows each query can retrieve from a database, specify a number of rows in the Max Rows column and change the Status Max Rows to either Enable, Warn, or Ignore:

- **Enable.** This limits the number of rows to the value specified. If the number of rows exceeds the Max Rows value, the query is terminated.
- **Warn.** 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. The Siebel Analytics Server administrator can use the information in the log entry to identify the query.
- **Ignore.** Limits will be inherited from the parent group. If there is no row limit to inherit, no limit is enforced.

To disable any limits, set the Status Max Row setting to Disable.

4 To specify the maximum time a query can run on the database, enter a time in the Max Time column and change the Status Max Time to either Enable, Warn or Ignore. These options are identical to those for limiting the number of rows returned.

5 To restrict access to a database during particular time periods, click on the button in the Restrict column to open the Restrictions dialog box.

- To explicitly allow access during a particular time period of the week, select the time period and click Allow.
- To explicitly disallow access during a particular time period of the week, select the time period and click Disallow.
- If a time period is not selected, access rights remain unchanged; if access is allowed or disallowed explicitly and in one or more groups, the user is granted the least restrictive access for time periods that are defined. 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.

6 To specify a filter to apply to an object, open the Filters tab of the User/Group Permissions dialog box.

7. Add an object to filter by clicking the Add button, navigating to the object you want to filter in the Browse dialog box, and double-clicking on the object. After selecting the object, it appears in the list in the User/Group Permissions Filters dialog box.

8. Click the Expression Builder button to the right of the Business Model Filter cell corresponding to the object you want to filter.

9. Use the Expression Editor to create a logical filter.

   The Status column indicates if the filter is in use and has the following values:

   - **Enable.** The filter is applied to any query that accesses the object.
   - **Disable.** 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.
   - **Ignore.** 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.

Using XML as a Data Source 16

This section describes the use of the Extensible Markup Language (XML) as a data source.

Overview

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 344
- "Using the Siebel Analytics Server XML Gateway" on page 346
- "Using XML ODBC" on page 364
- "XML Examples" on page 367

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 356 for more information.

URLs may include repository or session variables, providing support for HTTP data sources that accept user IDs and passwords embedded in the URL; for example: http://somewebserver/cgi.pl?userid=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 13, "Using Variables in a 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 219.

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 348).
- 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 358.

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 344.

   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 13, "Using Variables in a Repository."

4. Optionally, enter either an Extensible Stylesheet Language Transformations (XSLT) file or XPath expression.

   Use the Browse button to browse for XSLT source files.

5. Enter 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. Enter a name and optional description for the connection on the General tab. See "Working with Connection Pools" on page 121 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 268 in Chapter 11, "Query Caching."

    The default time-out interval for queries (URL loading time-out) is 15 minutes.

11. Click OK to complete the import.

12. For additional control over the XML data sources, you can specify an XSLT file or an XPath expression for individual tables in the data sources from the Physical Table dialog box. If specified, these entries are used to overwrite corresponding XSLT or XPath entries in the Connection Pool for the respective physical tables.

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>
            <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>
   <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
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

Column	Fully Qualified Name
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 367.

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 180.

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 27 provides a description of the API elements.

Table 27. 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 27. 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 Siebel Analytics Server Administration Tool, create a database and select XML Server as the database type.

For information about creating a database, see "Creating or Editing a Database Object" on page 118.

2 Configure the connection pool:

a Right-click the database you created in Step 1, and then select New Object > Connection Pool.

b Enter a name for the connection pool.

c Select XML as the call interface.

d Enter a data source name, and then click OK.

NOTE: Do not enter 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 358.

4 In the XML tab of the Physical Table dialog box, do one of the following:

- Select the Executable radio button, enter 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 enter 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>
```

SQL Reference 17

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 374
- "SELECT Usage Notes" on page 375
- "SELECT List Syntax" on page 376
- "Rules for Queries with Aggregate Functions" on page 377

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:

select_list	The list of columns specified in the query. See “SELECT List Syntax” on page 376.
from_clause	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 376.
search_condition	Specifies any combination of conditions to form a conditional test. See “WHERE Clause Syntax” on page 376.
column	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 377.

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 396
- "String Functions" on page 402
- "Math Functions" on page 411
- "Calendar Date/Time Functions" on page 418
- "Conversion Functions" on page 429
- "System Functions" on page 431
- "Expressing Literals" on page 431

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 381.

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 381.

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 381.

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 381.

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 381.

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 381.

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 381.

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 381.

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:

n_expression	Any expression that evaluates to a numerical value.
n	Any positive integer. Represents the average of the last n rows of data.

The MAVG function resets its values for each group in the query, according to the rules outlined in “Display Function Reset Behavior” on page 381.

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 nth 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 381.

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 381.

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 381.

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 381.

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 381.

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:

interval	The specified interval. Valid values are: SQL_TSI_SECOND SQL_TSI_MINUTE SQL_TSI_HOUR SQL_TSI_DAY SQL_TSI_WEEK SQL_TSI_MONTH SQL_TSI_QUARTER SQL_TSI_YEAR
integer_expression	Any expression that evaluates to an integer.
timestamp_expression	The timestamp used as the base in the calculation.

A null integer-expression or a null timestamp-expression passed to this function will result in a null return value.

In the simplest scenario, this function merely adds the specified integer value (integer-expression) to the appropriate component of the timestamp, based on the interval. Adding a week translates to adding seven days, and adding a quarter translates to adding three months. A negative integer value results in a subtraction (going back in time).

An overflow of the specified component (such as more than 60 seconds, 24 hours, twelve months, and so on) necessitates adding an appropriate amount to the next component. For example, when adding to the day component of a timestamp, this function considers overflow and takes into account the number of days in a particular month (including leap years when February has 29 days).

When adding to the month component of a timestamp, this function verifies that the resulting timestamp has a sufficient number of days for the day component. For example, blindly adding 1 month to '2000-05-31' results not 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;
```

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 1	The timestamp used in the function.
timestamp_expression 2	The first timestamp used in the function.

A null timestamp-expression parameter passed to this function will result in a null return value.

This function first determines the timestamp component that corresponds to the specified interval parameter. For example, SQL_TSI_DAY corresponds to the day component and SQL_TSI_MONTH corresponds to the month component.

The function then looks at the higher order components of both timestamps to calculate the total number of intervals for each timestamp. For example, if the specified interval corresponds to the month component, the function calculates the total number of months for each timestamp by adding the month component and twelve times the year component.

Finally, the function subtracts the first timestamp's total number of intervals from the second timestamp's total number of intervals.

The TimestampDiff function rounds up to the next integer whenever fractional intervals represent a crossing of an interval boundary. For example, the difference in years between '1999-12-31' and '2000-01-01' is one year because the fractional year represents a crossing from one year to the next (that is, 1999 to 2000). By contrast, the difference between '1999-01-01' and '1999-12-31' is zero years because the fractional interval falls entirely within a particular year (that is, 1999).

Microsoft's SQL Server exhibits the same rounding behavior; IBM's DB2 does not—it 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's 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.

The following is an example of the TimestampDiff function and its results:

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;
```

The Siebel Analytics Server pushes down the TIMESTAMPADD and TIMESTAMPDIFF functions to Microsoft's SQL Server and ODBC databases by default. While the Siebel Analytics Server can also push to IBM's DB2, the features table is turned off by default due to DB2's semantics. The features table is also turned off by default for Oracle, since Oracle databases do not fully support these functions.

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, enter 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, enter 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, enter 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, enter 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.

SQL Reference

Sample Scripts and Load Procedures for Usage Tracking Data

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 statistics are written to a usage tracking log file.

This section contains information and sample scripts to help you extract data from usage tracking log files and load it to appropriately formatted relational database tables.

For information about administering usage tracking, see "Administering Usage Tracking" on page 230.

Locating the Sample Scripts for Usage Tracking Data

The Create Table scripts referenced in this appendix are located in the Schema folder in the Siebel Analytics software installation folder.

Examples are given for Create Table scripts for DB2, SQL Server 7 and Oracle:

- SAACCT.Oracle.sql for Oracle
- SAACCT.DB2.sql for DB2
- SAACCT.MSSQL7.sql for SQL Server 7

NOTE: 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.

Extracting to Oracle

Before extracting usage tracking log file data, a table has to be created to store the data. The following is a sample script to create the table S_NQ_ACCT for Oracle 7 or 8 (SAACCT.Oracle.sql):

```
drop table S_NQ_ACCT;
create table S_NQ_ACCT (
UserName           varchar(128),
RepositoryName     varchar(128),
SubjectAreaName    varchar(128),
NodeId             varchar(15),
StartTimestamp     date,
StartDate          date,
StartHourMin       char(5),
EndTimestamp       date,
EndDate            date,
EndHourMin         char(5),
QueryText          varchar(1024),
SuccessFlag        integer,
RowCnt             integer default 0,
TotalTimeInSecs    integer default 0,
CompileTimeInSecs  integer default 0,
NumDBQueries       integer default 0,
CumDBTimeInSecs    integer default 0,
CumDBRows          integer default 0
);
```

The following is a sample SQL*Loader script for Oracle 7 or 8.

```
LOAD DATA
INFILE ‘D:\TEMP\NQAcct.20010214.071500.log’ BADFILE NQAcctng.bad
APPEND
INTO TABLE S_NQ_ACCT
fields terminated by ‘;’ optionally enclosed by '"'
(
   UserName,
   RepositoryName,
   SubjectAreaName,
   NodeId,
   StartTimestamp date (19) "YYYY-MM-DD HH24:MI:SS",
   StartDate date (10) "YYYY-MM-DD",
   StartHourMin,
   EndTimestamp date (19) "YYYY-MM-DD HH24:MI:SS",
   EndDate date (10) "YYYY-MM-DD",
   EndHourMin,
   QueryText char(1024),
   SuccessFlag integer external,
   RowCnt integer external,
   TotalTimeInSecs integer external,
   CompileTimeInSecs integer external,
   NumDBQueries integer external,
   CumDBTimeInSecs integer External,
   CumDBRows integer external
)
```

The next example shows a command to invoke the Oracle loader (for Oracle 7.3). Northwind/n is the user ID and password for logging on to Oracle, and ex1.ctl is the name of the SQL*Loader script. The user ID needs to have permission to update the S_NQ_ACCT table.

```
Sqlldr73 userid=northwind/n control=ex1.ctl
```

Extracting to DB2

Before extracting usage tracking log file data, a table has to be created to store the data. The following is a sample command script to create the table S_NQ_ACCT for DB2 UDB v6 (SAACCT.DB2.sql).

NOTE: StartTime and EndTime are defined as CHAR(19), because the DB2 load utility cannot read ODBC format (yyyy-mm-dd hh:mi:ss) for timestamps represented as character strings.

```
drop table S_NQ_ACCT
create table S_NQ_ACCT
(
   UserName          varchar(128) not null,
   RepositoryName    varchar(128) not null,
   SubjectAreaName   varchar(128) not null,
   NodeId            varchar(15) not null,
   StartTimestamp    char(19) not null,
   StartDate         date not null,
   StartHourMin      char(5) not null,
   EndTimestamp      char(19) not null,
   EndDate           date not null,
   EndHourMin        char(5) not null,
   QueryText         varchar(1024),
   SuccessFlag       integer not null,
   RowCnt            integer default 0,
   TotalTimeInSecs   integer default 0,
   CompileTimeInSecs integer default 0,
   NumDBQueries      integer default 0,
   CumDBTimeInSecs   integer default 0,
   CumDBRows         integer default 0
);
```

From the DB2 Command Line Processor, enter the following commands to load the table:

```
load from D:\TEMP\NQAcct.20010214.071500.log of DEL
   modified by coldel,
   Method P (1,2,3,4,5,6,7,8,9,10,11,12,13,14)
insert into S_NQ_ACCT(UserName, RepositoryName,
             SubjectAreaName, NodeId, StartTimestamp,
             StartDate, StartHourMin, EndTimestamp,
             EndDate, EndHourMin, QueryText, SuccessFlag,
             RowCnt, TotalTimeInSecs, CompileTimeInSecs,
             NumDBQueries, CumDBTimeInSecs, CumDBRows)
```

Extracting to SQL Server

Before extracting usage tracking log file data, a table needs to be created to store the data. The following is a sample command script to create the table S_NQ_ACCT for SQL Server 7 (SAACCT.MSSQL7.sql).

NOTE: StartTime and EndTime are defined as CHAR(19) due to SQL Server limitations with timestamps.

```
drop table S_NQ_ACCT
create table S_NQ_ACCT
(
   UserName           varchar(128),
   RepositoryName     varchar(128),
   SubjectAreaName    varchar(128),
   NodeId             varchar(15),
   StartTimestamp     char(19),
   StartDate          char(10),
   StartHourMin       char(5),
   EndTimestamp       char(19),
   EndDate            char(10),
   EndHourMin         char(5),
   QueryText          varchar(1024),
   SuccessFlag        integer,
   RowCnt             integer default 0,
   TotalTimeInSecs    integer default 0,
   CompileTimeInSecs  integer default 0,
   NumDBQueries       integer default 0,
   CumDBTimeInSecs    integer default 0,
   CumDBRows          integer default 0
);
```

The bulk copy utility (bcp) is used to load the usage tracking log files into the SQL Server database table. An example of the command used to invoke bcp follows (where /U is the database user ID, /P is the database password, /S is the name of the SQL Server machine, and /e is the name of the error output file):

```
bcp northwind.dbo.S_NQ_ACCT in D:\TEMP\NQAcct.20010214.071500.log /e
D:\TEMP\NQAcct.err /c/t;/r\n/Sserver1 /Usa /P
```

NOTE: Loaders may issue errors if the length of a string in the data file exceeds the maximum width of the string's corresponding column.

Description of the Usage Tracking Data

Table 28 describes each column in the usage tracking table.

Table 28. Usage Tracking Data

Column	Description
UserName	The name of the user who submitted the query.
RepositoryName	The name of the repository the query accesses.
BusinessModelName	The name of the business model being accessed.
NodeID	(Reserved for future use.)
Start timestamp	The date and time the logical query was submitted.
Start date	The date the logical query was submitted.
Start hourMin	The hour and minute the logical query was submitted.
End timestamp	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 date	The date the logical query was completed.
End hourMin	The hour and minute the logical query was completed.
QueryText	The SQL submitted for the query.
SuccessFlag	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.
RowCnt	The number of rows returned to the query client.
TotalTimeInSecs	The time in seconds required to compile and execute the query. This reflects only the time spent actually processing the query.
CompileTimeInSeconds	The time in seconds required to compile the query.

Table 28. Usage Tracking Data

Column	Description
NumDBQueries	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.
CumDBTimeInSecs	The total time in seconds spent computing back-end database queries. TotalTimeInSecs minus CumDBTimeInSecs equals the time spent within the Siebel Analytics Server.
CumDBRows	The total number of rows returned by the back-end databases.

Index

Symbols

.INI file
  NQClusterConfig.INI file, setting parameters for Cluster Server feature 301
  NQSConfig.INI file, collecting information about queries 235
  NQSConfig.INI file, tuning parameters 239
  NSQConfig.INI file, adding an entry 90
.log files
  See log files

A

Abs math function, about 411
Acos math function, about 411
Administration Tool
  See also Browse dialog box; logical business models; repository mode
  about 34
  Cache Manager, changing column order 72
  Cache Manager, selecting columns to display 72
  Edit menu, described 57
  File menu, described 56
  Help menu, described 58
  icons and symbols (table) 59
  join diagrams, setting default window size 72
  keyboard shortcuts (table) 59
  main window, repository parts described 55
  Manage menu, described and functions (table) 57
  preferences, setting general preferences 69
  repository components 79
  repository objects, adding or editing 74
  repository objects, setting permissions 73
  repository objects, specifying appearance in alphabetical order 71
  row counts, about updating for tables and columns 75
  row counts, turning on 75
  row counts, updating 75
  scrolling speed, setting 72
  Siebel Analytics Server, using to shut down 223
  toolbar functions 58
  Tools menu, described 58
  View menu, described 57
  Window menu, described 58
aggregate expression builder dialog
  See Expression Builder dialog boxes
aggregate fact table
  creating and example, about 202
  sources, creating for each logical table 203
aggregate functions
  about 386
  aggregate queries, about 377
  alternative syntax 383
  Avg aggregate function, about 387
  AvgDistinct, calculating average mean 387
  baseline columns, computing aggregates 377

BottomN, about ranking the lowest n values 387
Count (*) (CountStar), about counting number of rows 389
Count, about calculating the number of rows 388
CountDistinct, about adding distinct process to COUNT 388
display function, reset behavior 381
First, about selecting first returned value 389
GroupByColumn, about specifying logical columns 389
GroupByLevel, about specifying dimension levels 390
LAST, about selecting last returned value 390
Max, about calculating maximum value 391
measure columns, computing aggregates 379
Median, about calculating the median value of rows 391
Min, about calculating the minimum value 392
NTILE, about determining the rank of a value 392
Percentile, about calculating a percent rank 393
Rank, about calculating the rank for each value 393
StdDev, about returning standard deviation 394
Sum, about calculating 394
SumDistinct, about calculating sum by adding distinct values 395
TopN, about ranking the highest n values 395
aggregate navigation, setting up
See also aggregate tables; aggregate table fragments
aggregate levels, specifying for each source 200
aggregate table definitions, about and navigation 47
aggregate table fragments, about and example 211
aggregate tables, about 199
aggregated fact data, creating dimension sources for each level 202
aggregated fact data, creating sources for each logical table 203
dimension-specific aggregation rules, about and example 210
fragment content, specifying 204
WHERE clause filter, about and example 200
aggregate table fragments
about and example 211
about configuring a repository to use fragments 212
aggregate table content, specifying 212
physical joins for virtual table, constructing 215
physical layer table, defining with a Select Statement 212
physical layer table, example 214
SQL virtual table content, creating 215
aggregate tables
See also aggregate table fragments
about and navigation 199
aggregate table definitions, about and navigation 47
performance, about using to improve 239
aliases
Alias tab, using 115
synchronizing 184
analyzing the cache
See Cache Manager
application environment, about using metadata repositories to store information 34
architectural components
Administration Tool, about 34
diagram 33

metadata repositories, about 34
multithreaded architecture 34
Web client tools and ODBC-compliant tool, about 35
Web Server and Intelligence Dashboards, about 35
ASCII string function, about 402
Asin math function, about 411
Ask DBMS button, using to change Feature table entries 120
Atan math function, about 412
Atan2 math function, about 412
audience for guide 13
authentication cache, about disabling 326
authentication options
See also security; groups, working with
authentication, about 329
authentication, order of 337
database authentication, about using and procedure 335
external table authentication, about 332
external table authentication, setting up 333
LDAP authentication, about 330
LDAP authentication, setting up 331
logging level, setting 332
operating system authentication, about 329
operating system authentication, configuring (procedure) 330
password, changing 336
security, bypassing 337
Siebel Analytics Server user IDs and passwords, about and storage of 336
Siebel Server internal authentication, about 336
USER session system variable, defining for LDAP authentication 331
Avg aggregate function, about 387
AvgDistinct aggregate function, about 387

B

baseline column
behavior with aggregate functions 377
example 378
Between SQL logical operator, about 383
Bit_Length string function, about 403
BottomN aggregate function, about 387
bridge tables, using to model many-to-many relationships 51
Browse dialog box
about using 76
querying for an object 76
selecting an object in 76
synchronizing an object in the query results list with the tree control list 77
buffer size, configuring 247
Business Model and Mapping layer, working in
See also business models; logical table joins
about 149
business model, about working with 149
business model, creating (procedure) 150
dimensional hierarchy, working with 167
dimensions, about 165
dimensions, creating (procedure) 166
dimension-specific aggregation rules, setting up 158
hierarchies, about 165
logical columns, working with 155
logical table source 160
logical table sources (mappings), working with 160
logical tables, about working with 150
measure column, specifying a default aggregation rule 157
measure, associating with a level in dimension 160

measure, removing the association 160
physical to logical mapping, defining 162
repository, about setting up and working in 80
table content definitions, creating 164
business models
See also Business Model and Mapping layer, working in; presentation catalogs
about creating 86
consistency check, about passing 66
copy business model with presentation catalog utility 185
diagram, about using to create joins 175
diagram, displaying 178
repository, checking consistency within a 67
understanding 38

C

cache
behavior in offline mode 263
behavior in online mode 263
disabling 248
enabling 248
purging 266
purging when switching between repositories 263
cache authentication, about disabling 326
cache hits
description of 251
information about 251
cache information, viewing 309
Cache Manager
about and opening 264
column order, changing 72
columns, selecting columns to display 72
global cache information, displaying 265
menu option, described 57
option settings (table) 264
purging cache 266
view and save SQL call 264
cache persistence time setting 249
cache storage
cache data storage directories 246
cache entry, controlling max number of rows 247
cache metadata file, about 247
query caching, disabling query caching 248
query caching, enabling query caching 248
cache, monitoring and managing
disabling caching for the system, about and advantages 249
event polling tables, configuring 250
physical tables, about caching and persistence timing 249
Calculation Wizard, about 70
calendar date/time functions
Current_Date, about returning 418
Current_Time, about 419
Current_TimeStamp, about 419
Day_Of_Quarter, about 419
DayName, about 420
DayOfMonth, about 420
DayOfWeek, about 420
DayOfYear, about 421
Hour, about 421
Minute, about 421
Month, about 422
Month_Of_Quarter, about 422
MonthName, about 422
Now, about 423
Quarter_Of_Year, about 423
Second, about 423
TimestampAdd, about 423
TimeStampDiff, about 426
Week_Of_Quarter, about 428
Week_Of_Year, about 428
Year, about 428
Cast conversion function, about 429

catalog folders
creating or editing a catalog 128
Dynamic Name tab, sorting entries 129
Dynamic Name tab, specifying the session variable to use 129
Dynamic Name tab, unassigning a session variable 129
Ceiling math function, about 412
Char string function, about 403
Char_Length string function, about 403
character literals, about 431
checking in changes
changes, making available and saving to disk 68
Check In Changes dialog box, about using and tasks 67
checkpoints
described 234
client tools, connectivity to 273
cluster information, viewing
cache information, viewing 309
Cache view, about and columns described (table) 309
Request window (Session view), columns described (table) 311
Session window (Session view), columns described (table) 310
Session window, columns described (table) 310
Status view, about and columns described (table) 307
Cluster Manager
See also Cluster Server feature
accessing (procedure) 306
cache information, viewing 309
Cache view, about and columns described (table) 309
graphical user interface, described 306
managing clustered servers (procedure) 313
managing clustered servers, actions available 312
performance considerations 314
refreshing, setting and procedure 306
Session view, Request window (Session view) columns described (table) 311
Session view, Session and Request window, described 310
Session view, Session window columns described (table) 310
Status view, about and columns described (table) 307
stopped or offline, note about starting 306
using, about 306
Cluster Manager menu option, described 57
Cluster Server feature
See also Cluster Manager
about 297
cache information, viewing 309
Cluster Controller and Analytics Server, starting from the Command window 303
Cluster Controller and Analytics Server, starting manually in Windows 302
installing 300
managing clustered servers (procedure) 313
managing clustered servers, actions available 312
master server, about 298
NSCLusterConfig.INI file, about copying to Config directory 302
parameters, setting in the NQClusterConfig.INI file 301
parameters, setting in the NQSConfig.INI file 300
performance considerations 314
primary Cluster Controller, about 298
Repository Publishing Directory, about 299
secondary Cluster Controller, about 298
Siebel Analytics ODBC data source name, about configuring 302
startup process overview 304

stopped or offline, note about starting 306
column mapping
logical columns to physical columns, mapping 162
removing column mapping 163
Command window, stating Cluster Controller and Analytics Server 303
complex joins
about 141
logical complex joins, about and complex joins 175
logical complex joins, creating 177
Concat string function, about 404
conditional expressions
CASE (if), about and syntax 385
CASE (Switch), about and syntax 383
connection pool
about 80
creating and configuring, about 85
creating or editing 121
dialog box, fields described 122
connectivity
client tools and data sources, connectivity to 273
metadata, importing 274
ODBC data source names, configuring 269
query and reporting tools, about using 274
consistency check
about passing and example 66
business model within a repository, checking consistency of 67
repository, checking for consistency 66
conversion functions
Cast, about changing the data type to another data type 429
IfNull, about testing if an expression evaluates to a null value 429
VALUEOF(), about using the function in an expression builder 429
copy business model with presentation catalog utility, about and procedure 185
Cos math function, about 413
Cot math function, about 413
Count (*)/CountStar aggregate function, about 389
Count aggregate function, about 388
CountDistinct aggregate function, about 388
Current_Date calendar date/time function, about 418
Current_Time calendar date/time function, about 419
Current_TimeStamp calendar date/time function, about 419

D

Data Mining Adapter
See also XML Gateway; XML ODBC database type; XML, using as a data source
configuring (procedure) 362
In-Process Data Mining Adapter API, about 358
In-Process Data Mining Adapter API, sample implementation 360
In-Process Data Mining Adapter API, using ValueOf( ) expression 361
operation modes 358
data modeling
business model, understanding 38
logical business models, about 48
objectives of 37
physical database model, understanding 40
data sources
connectivity to 273
heterogeneous, about 48
data transformations, about 30
data warehouse, list of construction tasks 29

DATA_STORAGE_PATHS parameter, using to specify query cache storage 246
database hints
  See also databases
  database objects that accept objects (table) 145
  hints, creating (procedure) 147
  index hint, about 146
  Leading hint, about 146
  performance considerations 146
  SQL comment markers, about entering 147
  usage examples 145
  using, about 145
database object
  creating or editing in the Physical layer 118
  database hints, database objects that accept hints (table) 145
  ODBC type, about assigning if database type undetermined 118
Database system function, about 431
database type, restoring default entries for 120
databases
  See also database hints
  authentication, about using and procedure 335
  configuring, tuning, and indexing, importance 240
  database hints, about using 145
  symbol, described 59
datetime literals, about 432
Day_Of_Quarter calendar date/time function, about 419
DayName calendar date/time function, about 420
DayOfMonth calendar date/time function, about 420
DayOfWeek calendar date/time function, about 420
DayOfYear calendar date/time function, about 421
DB2
  usage tracking log files, extracting to 439
DCOM, changed client/server communication method 271
decimal literal, about 432
default client/server communication method, changed from DCOM to TCP/IP 271
Default initializer column, about value 295
Degrees math function, about 413
deleting
  alias 115
  column mapping 163
  drill path, deleting a level from 173
  foreign key 155
  initialization block 296
  key 154
  logical column 153
  logical table source 154
  measure, removing the association 160
  presentation tables 111
  presentation tables column 113
  table as a logical table source 162
description of 251
Diagnosis Records table, about limiting the number of records 53
dimensional hierarchies
  factual measures, about 51
  sample hierarchy rollups, about and diagram 50
  star and snowflake models, about 51
dimensional hierarchy
  See also individual dimensional entries
  about 167
  drill path, deleting a level from 173
  drill path, reordering a level 174
  general properties, defining for the level 172
  grand total levels, about 168
  grand total levels, example 169
  level attributes, about 167

level- based measure calculations, setting up 169
level keys, about 168
level-based measure calculations, about 169
level-based measure calculations, example 171
primary key for a level, adding 173
primary key, editing for a level 173
dimensional models
See also individual dimensional entries
about 40
bridge tables, using to model many-to-many relationships 51
sample hierarchy rollups, about and diagram 50
single table model, about and creating 54
understanding 49
dimensional schemas
about and advantages 41
start schema (diagram) 42
dimensions, defined and example 38
dimensions, working with
See also individual dimension entries
about 165
creating (procedure) 166
hierarchies, about 165
note, about including the key column in the dimension 166
dimension-specific aggregation rules
See also individual dimension entries
about and example 210
columns, specifying for 158
reordering aggregate rule 159
setting up, about 158
dirty data, about 30
display functions
example 382
reset behavior 381
DISPLAYNAME system session variable, about 282
dragging and dropping
logical tables 152
repository, setting up existing 83
drilling down
drill path, deleting a level from 173
drill path, reordering a level 174
driving table
about specifying 180
caution, about specifying when creating a Business Model Diagram 179
caution, about specifying when creating a logical complex join 178
caution, about specifying when creating a logical foreign key 177
caution, specifying and query optimization 181
controlling and tuning performance, about 180
logical joins, specifying (procedure) 181
DSN connection
See Siebel Analytics Server
Dynamic Name tab
entries, sorting 129
session variable, specifying 129
session variable, unassigning 129
dynamic repository variables
See also initialization blocks
about 279
example 279
initializing 286

E

Edit menu, described 57
EMAIL system session variable, about 283
entity-relationship (E-R) models
about 40
queries that perform historical analysis, performance of 41
environment diagram, high-level 27
event polling table
cache event processing, about 255
configuring 250

CREATE TABLE statements, sample event polling table 259
making the polling table active 260
overview 250
physical databases, setting up 256
populating, about 262
repository. making changes to 263
Siebel Analytics Event Table utility, identifying using 186
troubleshooting 262
using 262
Execute UDML utility
about and starting 186
repository, using to create an object in 97
Exp math function, about 413
Export logical keys option, about using with parameterized SQL queries 111
Expression Builder dialog boxes
about using 192
accessing 192
Aggregate Content folder, about 195
Constraints folder, about 195
Dimensions folder, about 195
example (diagram) 193
expression, building (procedure) 198
Expressions folder, about 195
Functions folder, about 195
Logical Tables folder, about 195
navigating within the expression builder 197
Operators folder, about 195
Repository Variables folder, about 196
Session Variables folder, about 196
setting up example 196
toolbar (table) 194
Types folder, about 195
expression literals
character literals, about and expressing 431
datetime literals, about and formats 432
decimal, about and expressing 432
floating point, about and expressing 433
integers, about and expressing 432
numeric literals, about 432
Extensible Markup Language
See XML, using as a data source
external table authentication
about 332
setting up 333
Externalize Strings utility, about and starting 187

F

fact table, about 43
facts (factual measures), defined and aggregation rule example 38
factual hierarchies, about dimensional hierarchies 51
Feature table entries, changing using Ask DBMS 120
File menu, described 56
filter
See also query execution privileges
complex filter, about constructing 99
constructing a filter to view all databases references in a business model 98
constructing a filter to view all Presentation layer columns mapped to a logical column 99
query results, constructing a filter for 98
First aggregate function, about 389
floating point literal, about 433
Floor math function, about 414
folder symbol, described 60
foreign keys
Foreign Keys tab, using to specify a driving table 180
logical foreign key, creating 176
note, about importing from an Oracle database 84
primary key, relationship with 140
fragmentation content
about 204
multicolumn content descriptions example 205

parallel content descriptions
example 206
single column range-based predicates
example 204
single column value-based predicates
example 204
unbalanced parallel content descriptions
example 209
fragmented data, about and example 139
FROM clause syntax, about 376
functions
See aggregate functions

G

grand total levels
about 168
example 169
granularity, defined 43
graphical user interface
See Cache Manager
GROUP BY clause
query behavior with and without 381
syntax, about 377
GROUP system session variable, about 282
GroupByColumn aggregate function,
about 389
GroupByLevel aggregate function,
about 390
groups, working with
See also authentication options
about 319
groups, about creating and example 320
LDAP authentication, configuring 325
LDAP, about importing users and
groups 325
member hierarchies, viewing in the Query
Repository dialog box 324
member hierarchies, viewing in the
Security Manager 324
predefined Administrators group,
about 320
privileges example (diagram) 322
privileges, about granting and
examples 320
repository, adding a new group to 323
repository, importing LDAP users and
groups into 327
user privileges and group privileges
example (diagram) 321
guide
audience for 13
organization of 14

H

Help menu, described 58
heterogeneous data sources, about 48
hierarchy, defined and example 38
historical time comparison, about and
starting Times Series Wizard 184
Hour calendar date/time function,
about 421
HTML tables
example 356
XML Gateway, accessing by 356

I

icons, described (table) 59
IfNull conversion function, about 429
importing
metadata 274
metadata, about connecting using an
ODBC connection 84
users and groups using LDAP 325
XML data using ODBC 364
In SQL logical operator, about 383
inconsistencies, about checking repository
or business model for
consistencies 66
indexing, about index hint
instructions 146
Initialization Block dialog box
accessing (procedure) 294
using, about 294

initialization blocks
about 286
block, linking with a variable not already refreshed 295
block, removing variable's association with 295
blocks, removing 296
caution, about opening Initialization Block dialog box in online mode 286
Default initializer column, about value 295
dynamic repository variables, initializing 286
initialization block execution order, setting 296
new variable, adding to refresh with block 294
note, about number of columns different from number retrieved 294
session variables, creating or editing (procedure) 290
session variables, initializing 288
Siebel Analytics Server, re-initializing when it starts 295
variable refreshed by this block, editing
variable, reordering 294
In-Process Data Mining Adapter API
about 358
column values, specifying 361
configuring (procedure) 362
sample implementation 360
ValueOf( ) expressions, using 361
Insert string function, about 404
integers literals, about 432
Is Null SQL logical operator, about 383

J

Job Manager menu option, described 57
join conditions, about defining on physical schema about defining on 85
join diagrams, setting default window size 72
Joins Manager
joins, about using to create 175
menu option, described 57

K

key symbol, described 60
keyboard shortcuts (table) 59

L

Last aggregate function, about 390
LDAP
See Lightweight Directory Access Protocol (LDAP)
Leading hint, about 146
Left string function, about 405
Length string function, about 405
Level dialog box
drill path, deleting a level from 173
drill path, reordering 174
primary key for a level, adding and editing 173
level key, about 168
level-based measure calculations
about 169
example 171
setting up 169
levels, working with
dimensional hierarchy, about 167
drill path, deleting a level from 173
drill path, reordering a level 174
general properties, defining for the level 172
grand total levels, about 168
grand total levels, example 169
level attributes, about 167
level keys, about 168
level-based measure calculations, about 169
level-based measure calculations, example 171
level-based measure calculations, setting up 169

primary key for a level, adding 173
primary key, editing for a level 173
Lightweight Directory Access Protocol (LDAP)
authentication, about 330
authentication, setting up 331
LDAP authentication, configuring 325
logging level, setting 332
passwords and storage, about 336
repository, importing LDAP users and groups into 327
USER session system variable, defining for LDAP authentication 331
users and groups, about using to import 325
Like SQL logical operator, about 383
literals, expressing (list of) 431
Load all objects option, about selecting 65
Locate string function, about 406
LocateN string function, about 406
log files
See also query log, administering; usage tracking, administering
NQServer.log and NQCluster.log, about opening and examining 303
Log math function, about 414
log viewer utility
log records, interpreting 228
running (procedure) 227
using, about 227
Log10 math function, about 414
logging levels
individual users, about enabling for 226
levels described (table) 226
log viewer utility, interpreting the log records 228
log viewer utility, using 227
user's logging levels
disabling 227
setting 227
logical business models
about 48
bridge tables, using to model many-to-many relationships 51
dimensional models, understanding 49
logical tables and columns and heterogeneous data sources 48
single table model, about and creating 54
Logical Column dialog box
logical column, unmapping from its source 157
logical table source, editing 157
measure, associating with a level in a dimension 160
measure, removing the association 160
logical columns
creating or editing 156
logical column, unmapping from its source 157
logical table source, editing 157
working with, about 155
logical complex join, creating 177
Logical Foreign Key dialog box, about using to specify a driving table 180
logical foreign key, creating 176
logical joins, specifying for driving table (procedure) 181
logical object, displaying all that map to physical tables 181
Logical Table dialog box
foreign key, deleting 155
foreign key, editing 155
key, deleting 154
key, editing 154
key, specifying 153
logical source, editing and existing 154
logical table source, deleting 154
logical table source, reordering 155
new logical table source, adding 154
logical table joins
See also Business Model and Mapping layer, working in
about 175

Business Model Diagram, displaying 178
creating, about 175
driving table, about specifying 180
driving table, about specifying and query optimization 181
driving table, controlling and tuning performance 180
driving table, specifying for logical joins (procedure) 181
logical complex join, creating 177
logical foreign key, creating 176
logical object, displaying all that map to 181
logical table sources
general properties, defining 161
removing a table as a source 162
Replace Wizard, using to replace tables or columns 185
Where clause filter, using to constrain physical tables 200
working with, about 160
logical tables
columns, about 48
creating and editing, ways to 151
creating by dragging and dropping 152
creating explicitly 151
foreign key, deleting 155
foreign key, editing 155
key, deleting 154
key, editing 154
key, specifying 153
logical column, adding to the column 152
logical column, deleting 153
logical column, editing 152
logical column, reordering 153
logical table source, deleting 154
logical table source, editing an existing 154
logical table source, reordering 155
new logical table source, adding 154
working with, about 150
LOGLEVEL system session variable, about 282
Lower string function, about 407

M

main window, repository parts described 55
Manage menu, described and functions (table) 57
many-to-many relationships, about using bridge tables 51
MASTER_SERVER parameter, about 298
math functions, list of 411
Mavg running aggregate function, about 396
Max aggregate function, about 391
measure
dimension, associating with 160
dimension, removing the association 160
measure column
behavior with aggregate functions 379
default aggregation rule, specifying a 157
example 380
Median aggregate function, about 391
member hierarchies
Query Repository dialog box, using to view 324
Security Manager, viewing in 324
menus, accessing context sensitive menus 81
metadata repository, about 34
metadata, importing
about 274
connections, about 84
Min aggregate function, about 392
Minute calendar date/time function, about 421
Mod math function, about 415
modes
See offline mode; online mode

Month calendar date/time function, about 422
Month_Of_Quarter calendar date/time function, about 422
MonthName calendar date/time function, about 422
More tab
joins diagrams, using to set default window size 72
scrolling speed, using to set 72
MSUM running aggregate function, about 397
multi-database joins, about 138
multithreaded architecture, about 34

N

Now calendar date/time function, about 423
NQClusterConfig.INI file, setting parameters for Cluster Server feature 301
nQLogViewer utility
log records, interpreting 228
query log file, using to view 227
NQQueryStats.log, about 235
NQSConfig.INI file
entry, adding an 90
queries, setting to collect more information about 235
tuning parameters, about 239
NQServer.log, and NQCluster.log, about opening and examining 303
NTile aggregate function, about 392
numeric literals, about 432

O

Object Type option, using to create virtual physical tables 131
Octet_Length string function, about 407
ODBC
calls from client applications, list of 275
client tools and data sources, about providing connectivity to 273
connection, about physical metadata imports 84
data source names, configuring 269
metadata, importing 274
query and reporting tools, about connecting with 274
offline mode
cache invalidation, implications for 263
repository, opening in 64
one-to-many relationships, about primary key-foreign key relationship (diagram) 45
online mode
cache invalidation, implications for 263
changes, making available and saving to disk 68
Check In Changes dialog box, about using and tasks 67
consistency check, about passing 66
consistency, checking repository for 66
repository, opening in 65
Options dialog box, using to set general preferences 69
Oracle database
note, about importing foreign keys from 84
usage tracking log files, extracting to 437
ORDER BY clause, about 377
organization of guide 14

P

parallel content descriptions, examples and discussion 207
password
changing 336
Siebel Analytics Server Administration account 318
Percentile aggregate function, about 393
performance
database hints, about resulting in better query performance 146
server configuration and tuning 239

permissions
See also privileges
adding or editing 74
repository objects, about setting for 73
physical column, creating or editing 136
physical database model, understanding
aggregate navigation, about setting up 47
contents of physical database, understanding 46
dimensional schemas, about and advantages 41
dimensional schemas, about star schema (diagram) 42
entity-relationship (E-R schemas, about 40
entity-relationship (E-R) schemas, performance for queries for historical analysis 41
physical models, types of 40
primary-key-foreign key relationships, about and diagram 45
Physical Diagram
command, about using 181
editor, about using to specify multi-database joins 138
physical joins
See also Physical layer, working in
about 138
complex joins, about 141
fragmented data, about and example 139
Joins Manager, using to define 141
multi-database joins, about 138
note, about imported key and foreign key joins 138
primary key and foreign key relationships, about 140
tip, about avoiding unnecessary joins 140
Physical layer, described 79
Physical layer, working in
See also physical joins
about 121
catalog, creating or editing 128
column mapping, removing 163
connection pool, creating or editing 121
database hints, about 145
database hints, database objects that accept hints (table) 145
database object, creating or editing 118
database type, restoring default entries for 120
Feature table entries, changing using Ask DBMS 120
logical columns, mapping to physical columns 162
permission, about setting 134
physical joins, defining in the Physical Table Diagram 143
physical joins, defining with the Joins Manager 141
physical layer objects, about 117
queries, specifying types of sent to a database 119
query type, locating 120
schema folders, working with 130
XML data source, setting properties for 126
physical schemas
importing (procedure) 84
importing, about 83
Physical Table Diagram
displaying 143
foreign key join or complex join, defining 143
physical joins, about using to define 143
physical tables
about 131
creating or editing 133
Physical Table dialog box, using Columns, Keys, and Foreign Keys tabs 135
virtual physical tables, creating using the Object Type option 131
XML data source, setting properties for 135

Pi math function, about 415
PORTALPATH system session variable, about 282
Position string function, about 407
Power math function, about 415
preferences
Cache Manager, changing column order 72
Cache Manager, selecting columns to display 72
general preferences, setting 69
join diagrams, setting default window size 72
repository objects, specifying appearance in alphabetical order 71
scrolling speed, setting 72
presentation catalogs
Alias tab, using 115
caution, about moving columns into presentation catalog folders 111
copy business model with presentation catalog utility 185
creating (procedure) 110
Presentation Catalog dialog box, described 110
presentation tables, deleting 111
table, reordering in the Presentation layer 112
tables, sorting in alphanumeric order 112
working in, about 109
presentation columns
Alias tab, using 115
creating or editing (procedure) 114
Presentation Column dialog box, described (table) 114
Repository Documentation utility, using to map to logical and physical columns 191
working with, about 113
Presentation Layer dialog box, using the Alias tab 115
Presentation layer, creating
See also Presentation layer, working in
business models, copying to the Presentation layer 87
columns, about removing unneeded or unwanted 87
creating, about 86
logical keys, about exporting in the Presentation Catalog 88
presentation columns, renaming 88
Presentation layer repository layer, about 81
presentation tables, renaming 87
Presentation layer, working in
See also Presentation layer, creating
Alias tab, using 115
Presentation Catalog dialog box, described 110
presentation catalogs, about working in 109
presentation catalogs, creating 110
Presentation Column dialog box, described (table) 114
presentation column, creating or editing 114
presentation column, deleting 113
presentation column, reordering 113
presentation columns, working with 113
Presentation layer, about and example 109
Presentation Tables dialog box, described (table) 112
presentation tables, creating 113
presentation tables, deleting 111
table, reordering 112
tables, sorting in alphanumeric order 112
presentation tables
Alias tab, using 115
column, deleting 113
creating (procedure) 113
presentation column, reordering 113

Presentation Tables dialog box, described (table) 112
primary key
foreign key, relationship with 140
specifying 136
primary key-foreign key relationship
about and diagram 45
PRIMARY_CONTROLLER parameter, about 298
privileges
See also permissions
explicit query privileges, specifying (procedure) 340
query privileges, about controlling and activities 339

Q

Quarter_Of_Year calendar date/time function, about 423
queries
aggregate functions, rules for 381
database, specifying types sent to 119
explicit query privileges (procedure) 340
Leading hint, about using to build the join order 146
query caching, enabling to improve performance 239
query privileges, controlling and activities 339
query type, locating 120
query caching
about 241
advantages of 241
architecture, about and diagram 245
cache event processing with an event polling table 255
cache hits 251
cache hits, information about 251
cache management strategy, choosing 249
Cache Manager, about and opening 264
Cache Manager, options settings (table) 264
cache storage, configuring 246
cache strategies 251
cost of caching, about 243
disabling for system 249
disabling query caching 248
enabling query caching 248
global cache information, displaying 265
improve performance, enabling to 239
invalidation after offline repository changes 263
note, about query references with different persistence times 134
parameters to control query caching, location of 241
purge cache options 266
purging cache 266
refresh interval, setting for XML data sources 268
security attribute, about enforcing 242
suite of queries, about running 254
query environment, administering
See also server configuration and tuning; usage tracking, administering
collecting more information, about 235
query log, administering 225
server, about connecting to 224
Siebel Analytics Server, shutting down in UNIX 223
Siebel Analytics Server, shutting down in Windows 221
Siebel Analytics Server, shutting down using the Administration Tool 223
Siebel Analytics Server, starting in UNIX 219
Siebel Analytics Server, starting in Windows 217
usage tracking, administering 230
query execution privileges 339
query log, administering
about 225

file size, controlling 225
log viewer utility, interpreting the log records 228
log viewer utility, using 227
logging levels
about enabling 226
described (table) 226
setting a user's 227
logging system, configuring 225
user's logging levels
disabling 227
Query Repository dialog box
about using 95
accessing 95
member hierarchies, using to view 324
new object, creating 96
object, reporting on and viewing the results online 97
parent of an object, seeing 96
Results list, editing an object in 97
Results pane, displaying fully qualified names 96
searching for object based on name 95
type of object, narrowing your search 96
Query Repository Filter dialog box
about and accessing 98
filter, constructing 98
filter, constructing to view all database references in a business model 98
filter, constructing to view all Presentation layer columns mapped to a logical column 99
query specification (SELECT statement), about 374
Query Statistics utility, about and starting 186
query tool, about connecting with 274

R

Radians math function, about 415
Rand math function, about 416
RandFromSeed math function, about 416
Rank aggregate function, about 393
RCOUNT running aggregate function, about 399
refresh interval, setting for XML data sources 268
Rename Wizard, about and starting 187
Repeat string function, about 408
Replace string function, about 408
Replace Wizard, about and starting 185
reporting tool, about connecting with 274
repositories
See also individual repository entries and privileges 92
Administration Tool, about using to create and edit repositories 34
comparing repositories 100
comparing, turning off Compare Mode 102
Execute UDML utility, using to create objects 186
LDAP authentication, configuring 325
LDAP, importing users and groups into 327
making changes to and implications for cache 263
merging repositories, about and process 103
merging version of Siebel Analytics Repository (procedure) 103
new group, adding to 323
new user, adding to 317
object in repository, creating using the Execute UDML utility window 97
repository metadata, managing 95
synchronizing 92
Repository Documentation utility, about and starting 191
Repository Import Wizard, about and using 92
repository mode
See also individual repository entries
business model within repository, checking consistency of 67

changes, making available and saving to disk 68
Check In Changes dialog box, about using and tasks 67
Load all objects option, about selecting 65
note, editing while repository is being loaded 64
offline mode, opening repository in 64
online mode, about passing consistency check 66
online mode, checking consistency of a repository 66
online mode, opening repository in 65
repository objects
See also individual repository entries
alphabetical order, specifying in 71
note, about naming objects Administrator 83
permissions, adding or editing 74
permissions, setting 73
repository variables
See also individual repository entries and initialization blocks
about 277
cache purging considerations 279
Default initializer column, about value 295
dynamic repository variables, about 279
dynamic repository variables, example 279
static repository variables, about 278
static repository variables, example 278
static repository variables, using in an expression 279
static repository variables, using in expression builder 278
uses for static repository variables 278
repository, managing metadata
See also individual repository entries and Query Repository dialog box
complex filter, about constructing 99
filter, constructing to view all Presentation layer columns mapped to a logical column 99
note, about constructing more than one filter 99
query results, constructing a filter for 98
repository, setting up
See also individual repository entries
Administration Tools, repository components in 79
business model, about creating 86
connection pool, about creating and configuring 85
context sensitive menu, accessing 81
data source, about defining 91
existing repository, dragging and dropping 83
join conditions, about defining on the physical schema 85
new repository, creating 83
NQSConfig.INI file, adding entry 90
physical schemas, about importing 83
physical schemas, importing (procedure) 84
Presentation layer, about creating 86
saving, checking consistency, and correcting errors 88
setup checklist (table) 82
Siebel Analytics Server, about starting 91
testing and refining, about 91
user community, about publishing to 91
REPOSITORY_PUBLISHING_DIRECTORY parameter, about 299
REQUESTKEY system session variable, about 283
Right string function, about 408
RMAX running aggregate function, about 400
RMIN running aggregate function, about 401
rollover, described 235
Round math function, about 416

row count feature
turning on 75
updating for tables and columns 75
updating the row counts 75
Row-Wise Initialization feature
about and example 288
variable with a List of Values, initializing 289
RSUM running aggregate function, about 398
running aggregate functions
about 396
Mavg, about calculating a moving average 396
MSUM, about calculating a moving sum 397
RCOUNT, about counting number of input records 399
RMAX, about showing the maximum values of a set of records 400
RMIN, about showing the minimum values based on a set of records 401
RSUM, about calculating a running sum 398

S

sample scripts, locating and example 436
schema folders
See also schemas
schema name, using a variable to specify 130
schema object, creating 130
schemas
See also schema folders
physical schemas, about importing 83
physical schemas, importing (procedure) 84
scrolling speed, setting 72
Second calendar date/time function, about 423
SECONDARY_CONTROLLER parameter, about 298
security
See also authentication options; privileges
bypassing 337
group privileges, about granting and examples 320
groups, about creating and example 320
groups, about working with 319
LDAP authentication, configuring 325
LDAP, about importing users and groups 325
member hierarchies, viewing in the Query Repository dialog box 324
member hierarchies, viewing in the Security Manager 324
predefined administrators group 320
privileges example (diagram) 322
repository, adding a new group to 323
repository, adding new user to 317
repository, importing LDAP users and groups into 327
Siebel Analytics Server Administration account password 318
Siebel Analytics Server Administrator account, about 318
Siebel Analytics Server Security, about 81
user accounts, about working with 316
user privileges and group privileges example (diagram) 321
Security Manager
See also security
member hierarchies, using to view 324
menu option, described 57
SELECT list syntax
about and syntax 376
FROM clause syntax, about 376
GROUP BY clause syntax, about 377
ORDER BY clause syntax, about 377
WHERE clause syntax, about 376
SELECT statement
about and basic syntax 374
conditional expressions, list of 383

queries and aggregate functions, rules for 377
query specification 374
Select list syntax 376
SQL logical operators 383
usage notes 375
WHERE clause 376
server configuration and tuning
See also query environment, administering
aggregate tables, about using 239
databases, importance of configuring, tuning, and indexing 240
NQSConfig.INI parameters. about using for tuning 239
query caching, about enabling to improve performance 239
server session management
See server configuration and tuning; Session Manager
Session Manager
See also query environment, administering
active query, killing 238
disconnecting a user from a session 238
menu option, described 57
opening (procedure) 238
Request Window fields (table) 237
Session Window fields (table) 237
update speed, controlling 238
using, about 236
session variables
See also initialization blocks
about 277
creating or editing (procedure) 290
initializing, about 288
nonsystem session variables, about using 283
row-wise initialization, about and example 288
system session variables, about using 281
system session variables, table of 282
using for authenticating users 281
shutting down Siebel Analytics Server
Administration Tool, using to shut down server 223
UNIX, shutting down server 223
Windows, shutting down from a command prompt 222
Windows, shutting down from the Services applet 221
Siebel Analytics Client testing tool 35
Siebel Analytics Event Tables utility, about and starting 186
Siebel Analytics Scheduler, about 57
Siebel Analytics Server
See also Cluster Server feature
Administration Tool, shutting down using 223
automatic startup, configuring for 218
business model, about interface to physical databases 40
connecting to, about 224
fails to start 220
internal authentication, about 336
nonlocal files, about accessing 345
note, about changing repository in online mode and attempting to stop server 68
password, changing 336
repository variable initialization blocks, re-initializing 295
UNIX, running the shutdown script 223
UNIX, running the Siebel startup script 219
user ID, changing 219
user IDs and passwords, about and storage of 336
Windows, configuring for automatic startup 218
Windows, shutting down from a Windows command prompt 222
Windows, shutting down from the Services applet 221

Windows, starting from the Services applet 217
Siebel Analytics Server Administrator
account, about and password 318
group, about 320
Siebel Analytics Server Security, about 81
Siebel Analytics Server XML Gateway
See XML Gateway
Sign math function, about 417
Sin math function, about 417
single table model, about and creating 54
SKIN system session variable, about 283
snowflake models, about dimensional hierarchies 51
Sort Objects tab, using to specify repository objects in alphabetical order 71
SQL 92 functions
datetime literals, about 432
NTILE function, about 392
SQL FROM clause syntax, about 376
SQL functions
aggregate functions, about 386
calendar date/time functions, about 418
conversion functions, about 429
datetime literals, about 432
expression literals, about 431
NTILE function, about 392
running aggregate functions 396
string functions, about 402
system functions, about 431
SQL logical operators, list of 383
SQL queries, about selecting the Export logical keys option 88
SQL Server, extracting user tracking log files to 441
SQL statement
database hints, about 145
database hints, creating 147
database objects that accept hints (table) 145
SQL syntax and semantics
conditional expressions, list of 383
queries and aggregate functions, rules for 377
Select list syntax 376
Select statement, about and basic syntax 374
Select usage notes 375
SQL logical operators 383
SQL WHERE clause syntax, about 376
Sqrt math function, about 417
star schema
about and diagram 42
dimensional hierarchies, about 51
static repository variables
See also initialization blocks
about 278
example 278
expression builders, using in 278
expression, using in 279
status bar, showing 71
StdDev aggregate function, about 394
STORAGE_DIRECTORY parameter
user tracking log files, selecting an output location for 230
string functions
about 402
Abs, about calculating the absolute value 411
Acos, calculates the acr cosine of a numerical expression 411
ASCII, about converter single character string to 402
Asin, about calculating the arc sine of a numerical expression 411
Atan, about calculating the arc tangent of a numerical expression 412
Atan2, about calculating the arc tangent of y/x 412
Bit_Length, about returning length in bits 403
Ceiling, about rounding a noninteger numerical expression 412
Char, about converting a numerical value 403

Char_Length, about returning length in number of characters 403
Concat, about forms of function 404
Cos, about calculating the cosine of a numerical expression 413
Cot, about calculating the cotangent of a numerical expression 413
Degrees, about converting an expression from radians to degrees 413
Exp, about sending the value e to the power specified 413
Floor, about rounding a noninteger numerical expression 414
Insert, about inserting a character string 404
Left, about returning characters from the left of a string 405
Length, about returning the length in number of characters 405
Locate, about returning the numerical position of the character_expression1 406
LocateN, about returning the numerical position of the character_expression1 406
Log, calculated the natural logarithm of an expression 414
Log110, about calculating the base 10 logarithm of an expression 414
Lower, about converting a character string to lower case 407
Mod, about dividing the first numerical expression 415
Octel_Length. about returning the bits in base 8 units 407
Pi, about returning the value of pi 415
Position, about returning the numerical position of the character_expression1 407
Power, about taking the first numerical expression to the power specified 415
Radians, about converting from degrees to radians 415
Rand, about returning a pseudo-random number 416
RandFromSeed, about returning a pseudo-random number from a seed value 416
Repeat, returns a specified expression n times 408
Replace, about replacing specified characters 408
Right, about returning a specified number of characters from the right of the string 408
Round, about rounding a numerical expression to n digits 416
Sign, about returning a value of 1, -1, or 0 417
Sin, calculated the sine of a numerical expression 417
Sqrt, about calculating the square root of the numerical expression argument 417
Substring, about creating a new string starting from a fixed number 409
Tan, about calculates the tangent of a numerical expression 418
TrimBoth, about stripping specified leading and trailing characters 409
TrimLeading, about stripping specified leading characters 410
TrimTrailing, about stripping specified trailing characters 410
Truncate, about truncating a decimal number 418
Upper, about converting a character string to uppercase 410
Substring string function, about 409
Sum aggregate function, about 394
SumDistinct aggregate function, about 395
symbols, described (table) 59

Synchronize Aliases utility, about and procedure 184
system
  session variables, about and LDAP authentication 331
  SQL functions, about 431
  variables, about and external table authentication 333

T

tables
  content definitions, creating 164
  event polling table, identifying 186
  one-to-many relationship, about and diagram 45
Tan math function, about 418
TCP/IP, client/server communication method changed 271
text strings, using the Externalize Strings utility to translate 187
Times Series Wizard, about and starting 184
TimestampAdd calendar date/time function, about 423
TimeStampDiff calendar date/time function, about 426
toolbar
  docking 58
  on/off, toggling 58
Tools menu, described 58
TopN aggregate function, about 395
TrimBoth string function, about 409
TrimLeading string function, about 410
TrimTrailing string function, about 410
troubleshooting
  event polling table 262
  Siebel Analytics Server fails to start 220
Truncate math function, about 418
Turn Off Compare Mode, enabling 102

U

UDML utility, starting 186
Unified Logon, about support of 329
Universal Database Markup Language, using the Execute UDML utility to create objects 186
UNIX
  shutting down the Siebel Analytics Server 223
  starting the Siebel Analytics Server 219
Update Physical Layer wizard, about and starting 188
Upper string function, about 410
usage tracking log files
  See also usage tracking, administering
  DB2 extracting to 439
  Oracle, extracting to 437
  sample scripts, locating and example 436
  SQL Server, extracting to 441
  usage tracking data (table) 443
usage tracking, administering
  See also Session Manager; usage tracking log files
  about and example 230
  file naming conventions, about and example 231
  note, about error accessing usage tracking output file 231
  output file, about and schema described (table) 231
  output file, column behavior in 233
  output file, format of 231
  output location, selecting 230
  performance considerations, about and checkpoints described 234
  queries, collecting more information about 235
user community, about publishing to 91
user ID, changing for the Siebel Analytics Server 219

user interface components
Edit menu, described 57
File menu, described 56
Help menu, described 58
icons and symbols (table) 59
keyboard shortcuts (table) 59
main window, described 55
Manage menu, described 57
toolbar functions, described 58
Tools menu, described 58
View menu, described 57
Window menu, described 58
USER session system variable, defining for LDAP authentication 331
User system function, about 431
USER system session variable, about 282
USER_LOG_FILE_SIZE parameter, using to control query log file size 225
users
See also privileges
LDAP, using to import users 325
new user, adding to repository 317
Siebel Analytics Administration account, about and password 318
user accounts, about 316
utilities
copy business model with presentation catalog utility, about and procedure 185
Execute UDML utility, about and starting 186
Externalize Strings utility, about and starting 187
log viewer utility, interpreting the log records 228
log viewer utility, using 227
Query Statistics utility, about and starting 186
Repository Documentation utility, about and starting 191
Siebel Analytics Event Tables utility, about and starting 186
Synchronize Aliases, about and procedure 184

V

VALUEOF( ) conversion function, about 429
ValueOf( ) expressions, using 361
Variables Manager menu option, described 57
variables, using
See also Initialization Block dialog box; initialization blocks
Default initializer column, about value 295
dynamic repository variables, about and example 279
new variables, creating 284
nonsystem session variables, about using 283
session variables, about 281
Siebel Analytics Server, re-initializing when it starts 295
static repository variables, about and example 278
static repository variables, using variables in expression builders 278
system session variables, about and LDAP authentication 331
system session variables, about using 281
system session variables, table of 282
system variables, about and external table authentication 333
Variable Manager, about and classes of variable 277
Variable Manager, accessing 277
View menu, described 57
virtual physical tables, creating using the Object Type option 131

W

WEBGROUPS system session variable, about 282
Week_Of_Quarter calendar date/time function, about 428
Week_Of_Year calendar date/time function, about 428
WHERE clause
  filter, about and example 200
  syntax, about 376
Window menu, described 58
Windows
  Cluster Controller and Analytics Server, starting manually 302
  NT and 2000 Administrator account, about and Siebel Analytics Server Administrator account 318
  NT and 2000, about support of Unified Logon 329
  NT and 2000, user ID to access nonlocal files 345
  ODBC data source names, configuring 269
  shutting down Siebel Analytics Server from a command prompt 222
  shutting down Siebel Analytics Server from the Services applet 221
  Siebel Analytics Server, configuring for automatic startup 218
  starting Siebel Analytics Server from the Services applet 217
wizards
  Calculation wizard, about 70
  Rename Wizard, about and starting 187
  Replace Wizard, about and starting 185
  Repository Import Wizard, about and using 92
  Time Series Wizard, about and starting 184
  Update Physical Layer Wizard, about and starting 188
workspace, Administration Tool
  Logical Table dialog box, using 154
  Presentation Catalog dialog box, about using Presentation Table tab 110
  Presentation Tables dialog box, about using Columns tab 112
  Rename Wizard, using to rename items 187

X

XML data source
  Ask DBMS, about availability of 120
  connection properties, setting 127
  physical table, setting properties for 135
  properties, about setting 126
  query output format settings, specifying 127
  refresh interval, setting 268
XML Gateway
  See also Data Mining Adapter; XML ODBC database type; XML Gateway
  about using 346
  example 348
  example, more complex 351
  HTML tables, accessing 356
  HTML tables, example 356
  XML data, importing using the XML Gateway 347
  XML examples 367
XML ODBC database type
  See also Data Mining Adapter; XML ODBC database type; XML Gateway
  example 365
  importing XML data 364
  XML data sources, about accessing 364
  XML examples 367
XML, using as a data source
  See also Data Mining Adapter; XML ODBC database type
  about 343
  example, more complex 351
  HTML tables, accessing 356
  HTML tables, example 356

importing data using the XML Gateway 347
XML examples 367
XML Gateway example 348
XML Gateway, about using 346
XML URL, locating 344
XPath expressions, support of 345
XSL transformation files (XSLT), support of 345

Y

Year calendar date/time function, about 428