

JASPERSOFT ULTIMATE GUIDE



Jaspersoft OLAP 4.0

ULTIMATE GUIDE

Copyright © 2011 Jaspersoft Corporation. All rights reserved. Printed in the U.S.A. Jaspersoft, the Jaspersoft logo, JasperAnalysis, JasperServer, JasperETL, JasperReports, JasperReports Server, and iReport, are trademarks and/or registered trademarks of Jaspersoft Corporation in the United States and in jurisdictions throughout the world. All other company and product names are or may be trade names or trademarks of their respective owners.

This is version 0311-JSP40-8 of the *Jaspersoft OLAP Ultimate Guide*.

TABLE OF CONTENTS

Chapter 1 Introduction	7
1.1 Welcome to Jaspersoft OLAP	7
1.2 Community and Commercial Editions	7
1.3 About this Guide	8
1.3.1 Technical Business Analyst	8
1.3.2 System Developer	8
1.3.3 System Administrator and Database Administrator	8
1.4 Other Resources	8
1.4.1 Standard Documentation	8
1.4.2 Premium Documentation	9
1.4.3 JasperForge	9
Chapter 2 Getting Started	11
2.1 On-Line Analytical Processing	11
2.1.1 OLAP Schema	12
2.1.2 Cubes	12
2.1.3 Dimensions	12
2.1.4 Measures and Facts	13
2.2 Jaspersoft OLAP and the Repository	13
2.3 Creating a Jaspersoft OLAP Environment	14
2.3.1 Creating the Database	15
2.3.2 Importing Data using an SQL Database Script	16
2.3.3 Defining a Data Source in JasperReports Server	17
2.3.4 Creating and Uploading an OLAP Schema	18
2.3.5 Creating a Mondrian Connection	20
2.3.6 Defining an MDX Query	24
2.3.7 Configuring XML/A	25
Chapter 3 Analyzing Data in a View	27
3.1 OLAP Views	27

3.2	Analyzing Data in an OLAP View	30
3.2.1	Displaying Product Sales by Quarter	31
3.2.2	Displaying Product Sales by Time across Store Locations	34
3.2.3	Sorting the Display	35
3.2.4	Drilling Through to Details	36
3.2.5	Displaying Charts	37
3.2.6	Exporting Output	38
3.3	Using the OLAP View Tools	39
3.3.1	Analysis Tool Bar and Navigation Table	39
3.3.2	Cube Configuration	40
3.3.3	Dimension Column Layout	46
3.3.4	Navigation Table Controls	48
3.3.5	Charts	51
3.3.6	Output Formats	53
3.3.7	Save Options	54
3.4	Troubleshooting Jaspersoft OLAP	54
3.4.1	Common Error Messages	54
3.4.2	Messages that Occur When Creating and Modifying OLAP Views	56
3.4.3	Error Messages When Creating Other Objects	57
3.4.4	Messages that Occur When Loading OLAP Views	58
3.4.5	Messages that Occur When Expanding Hierarchy Positions and Members	61
3.4.6	Error Messages that Occur When Drilling Through	62
Chapter 4	Securing Data in Jaspersoft OLAP	63
4.1	Securing Jaspersoft OLAP Data: A Business Case	63
4.2	Overview of CZS's Process	64
4.3	Understanding Access Grant Definitions and Profile Attributes	65
4.3.1	Access Grant Definitions	65
4.3.2	Profile Attributes and Variable Substitution	67
4.4	Configuring CZS's OLAP view	67
4.4.1	Defining a Sales Numbers Cube	67
4.4.2	Creating an OLAP Schema	68
4.4.3	Determining Roles	68
4.4.4	Selecting Users for the Roles	69
4.4.5	Assigning Profile Attributes	70
4.4.6	Creating a Mondrian Connection	71
4.4.7	Generating a Sales OLAP View	72
4.4.8	Testing the Results	72
4.5	Reference Material	74
4.5.1	OLAP Schema	74
4.5.2	Access Grant Definition	75
Chapter 5	Administering Jaspersoft OLAP	77
5.1	Understanding the Design Concepts	77
5.1.1	Introduction	77

5.1.2	Designing the Model	78
5.1.3	Applying the Model	79
5.2	Maintaining the System	85
5.2.1	XML/A Definitions	85
5.2.2	Monitoring the System	88
5.2.3	Maintaining Tables	88
5.2.4	Clearing the Cache	89
5.3	Tuning Performance	90
5.3.1	Understand Jaspersoft OLAP Performance	90
5.3.2	Profiling Performance	91
5.3.3	Jaspersoft OLAP Tuning Process and Options	92
5.4	Integrating Jaspersoft OLAP in the Enterprise's Data Flow	96
5.5	Troubleshooting Information for Technical Support	98
Chapter 6	Executing OLAP Views Using the HTTP Interface	99
6.1	Syntax	99
6.2	Example HTTP Calls for Jaspersoft OLAP	100
Appendix A	Example Foodmart Schemas	101
A.1	FoodMart Database Schema	102
A.2	FoodMart Sales OLAP Schema	103
Glossary		105
Index		113

CHAPTER 1 INTRODUCTION

Jaspersoft OLAP helps your users make well-informed decisions by providing advanced analysis tools.

This chapter has these sections:

- [Welcome to Jaspersoft OLAP](#)
- [Community and Commercial Editions](#)
- [About this Guide](#)
- [Other Resources](#)

1.1 Welcome to Jaspersoft OLAP

Jaspersoft OLAP runs within JasperReports Server. JasperReports Server itself builds on JasperReports, the world's most popular open source Java reporting library. It provides a comprehensive family of Business Intelligence (BI) products, including robust static and interactive reporting, report server, data analysis, and data integration capabilities. You can use Jaspersoft OLAP as a stand-alone product or as part of an integrated end-to-end BI suite that utilizes common metadata and provides shared services, such as a repository, security, and scheduling. JasperReports Server exposes comprehensive public integration interfaces enabling seamless embedding into other applications as well as the capability to easily add custom functionality.

1.2 Community and Commercial Editions

Jaspersoft OLAP is a component of our community project and commercial offerings. Each edition integrates the standard features of JPivot and Mondrian with additional features, including an enhanced user interface, streamlined tool bars and icons, fast expand/collapse features, and consistent option panes. The commercial editions also include row-level data security, performance tuning, and profiling reports and views to identify optimization candidates.

This guide discusses all editions. Sections of the guide that apply only to the commercial editions are indicated with a special note.

1.3 About this Guide

Because this ultimate guide is a comprehensive resource for users with many different needs, it includes information that may not apply to you or to your edition of Jaspersoft OLAP. The following audience descriptions and document maps can help you find the information that is most important to you.

1.3.1 Technical Business Analyst

Technical Business Analysts know their business, data, and processes; they are power users who generate business intelligence for others.

If you're a Technical Business Analyst, refer to the following sections of this document:

- [2.1, “On-Line Analytical Processing,” on page 11](#)
- [Chapter 3, “Analyzing Data in a View,” on page 27](#)
- [Chapter 4, “Securing Data in Jaspersoft OLAP,” on page 63](#)
- [5.1, “Understanding the Design Concepts,” on page 77](#)

1.3.2 System Developer

System Developers leverage Jaspersoft OLAP functionality in their own product. They extend and change its code, system configurations, and other low-level options.

If you're a System Developer, refer to the following sections of this document:

- [2.1, “On-Line Analytical Processing,” on page 11](#)
- [Chapter 3, “Analyzing Data in a View,” on page 27](#)
- [5.4, “Integrating Jaspersoft OLAP in the Enterprise’s Data Flow,” on page 96](#)
- [Appendix A, “Example Foodmart Schemas,” on page 101](#)

1.3.3 System Administrator and Database Administrator

System Administrators install, deploy, maintain, and troubleshoot Jaspersoft OLAP along with other systems in their environment.

Database Administrators (DBAs) administer database management systems (DBMS), and are familiar with both relational and On-Line Analytical Processing (OLAP) databases. They plan, configure, tune, and maintain the schemas that store business data.

If you're a System Administrator or DBA, refer to the following sections of this document:

- [Chapter 2, “Getting Started,” on page 11](#)
- [Chapter 3, “Analyzing Data in a View,” on page 27](#)
- [Chapter 5, “Administering Jaspersoft OLAP,” on page 77](#)
- [Appendix A, “Example Foodmart Schemas,” on page 101](#)

1.4 Other Resources

1.4.1 Standard Documentation

Jaspersoft OLAP includes standard documentation, such as build, install, and user guides. The documentation is found in the distribution image from which you installed the application. The documentation for the commercial editions is also available from Technical Support, while the Community Edition documentation is also available at JasperForge.org.

1.4.2 Premium Documentation

Some sections of this document reference the other Jaspersoft premium guides:

- *JasperReports Server Ultimate Guide*
- *JasperReports Server External Authentication Cookbook*
- *Jaspersoft OLAP Ultimate Guide*
- *iReport Designer Ultimate Guide*

Commercial Jaspersoft OLAP users can get these documents from Technical Support. For community users, the documents are available for purchase, separately or in documentation packs, at <http://www.jaspersoft.com/ultimate-guides>.

1.4.3 JasperForge

JasperForge.org is the online site for the open source version of Jaspersoft BI Suite. The suite is our open source solution developed by the Business Intelligence community and is accessible to anyone who needs to make well-informed decisions. Visit JasperForge.org for further resources and forum discussions.

CHAPTER 2 GETTING STARTED

This chapter is an overview of the concepts that underlie Jaspersoft OLAP. It also explains how to prepare data for analysis. For information about logging in on JasperReports Server and accessing Jaspersoft OLAP, refer to the *JasperReports Server Ultimate Guide*.

The chapter has these sections:

- **On-Line Analytical Processing**
- **Jaspersoft OLAP and the Repository**
- **Creating a Jaspersoft OLAP Environment**

2.1 On-Line Analytical Processing

On-Line Analytical Processing (OLAP) entails analyzing quantitative and categorical data using a set of analytical operations. We compare and contrast the quantitative data (measures) among a set of categories (dimensions). Jaspersoft OLAP facilitates such analysis with a standard set of functions, often called operations. OLAP compliments traditional reporting with a series of dynamic reports. Analytical operations let you filter information, get instant and incremental feedback, and devise what-if questions across a set of categories for comparing various numeric data. OLAP is well suited to sales analysis, customer profiling, and trend analysis.

Generally, the people most interested in OLAP applications are technical business analysts with in-depth knowledge of their data and basic capability in analysis. From their perspective, OLAP systems answer five questions:

- Who buys the product?
- What products are they buying?
- Where are the most successful stores?
- When are sales being made?
- Why are certain products selling, and under which promotional conditions?

You can derive the answers to these questions from a special data structure known as a cube. A cube contains all the data pertaining to a single business context, such as sales activities and customer demographics. To analyze the data in the cube, use the Jaspersoft OLAP web interface to slice and dice, expand and collapse, zoom in and out, and drill-through. For more information on these operations, see [3.3, “Using the OLAP View Tools,” on page 39](#).



The examples in this guide assume that you are using MySQL. If you are using a different database, you may need to use slightly different MDX queries; for example, if you use Oracle, you cannot use the upper-case member names shown in the examples.

2.1.1 OLAP Schema

An OLAP schema is a logical model that defines a multidimensional data structure. It defines one or more cubes in a single database that each are defined by one or more dimensions and measures. In Jaspersoft OLAP, the schema maps OLAP concepts onto the underlying database tables and columns, and also defines calculated fields and other OLAP relationships that don't exist in the physical database.

The Jaspersoft OLAP Workbench is a graphical tool to help you define and test the schema. You can also edit the schema XML file in a text editor.

Because Jaspersoft OLAP relies on Mondrian (an OLAP engine), the OLAP schemas you create for Jaspersoft OLAP must follow the Mondrian schema format, which is written in XML (for an example, refer to section A.2, “**FoodMart Sales OLAP Schema,**” on page 103).

There are many graphical tools that can help you build a schema, including the Workbench.

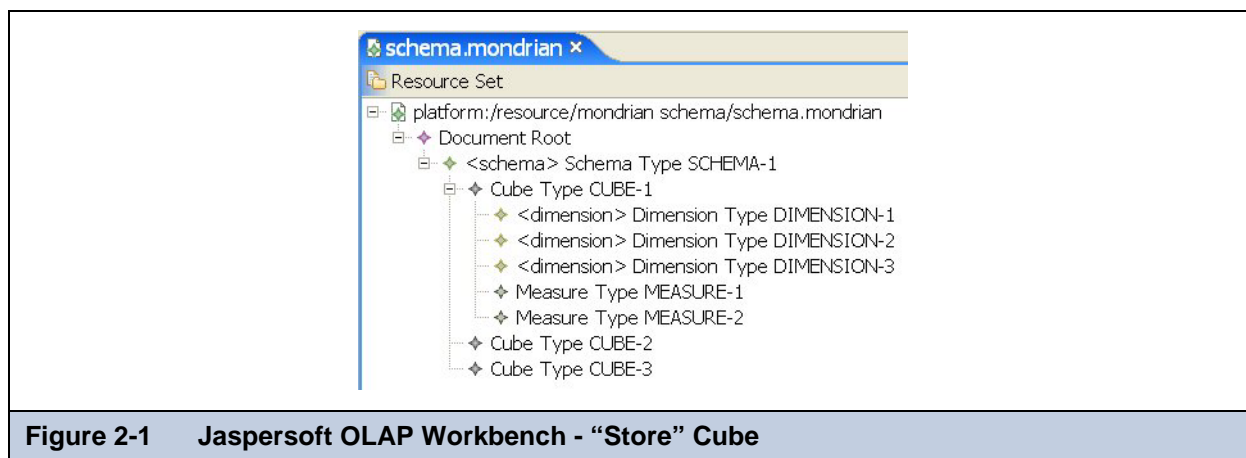


Figure 2-1 Jaspersoft OLAP Workbench - “Store” Cube

The Workbench enables you to define a cube, including its dimensions, measures, and member properties, using data stored in a JDBC data source. The Workbench is freely available from: <http://sourceforge.net/projects/jasperserver/files/JasperServer/>.

For information regarding the format that OLAP schemas must follow, refer to the *Mondrian Technical Guide*, which is included with the [JasperReports Server documentation](#).

2.1.2 Cubes

A cube is a multidimensional data structure containing dimensions (which encapsulate the characteristics) and measures (quantitative information) that describe a logical, connected set of information.

For example, **Figure 2-1** depicts a cube for sales by product and by store; the cube presents the sales figures (quantitative data) for a range of products and a group of stores (characteristics). The sales figures, such as STORE SALES and SALES COUNT, are the measures, and the product categories and store locations are the dimensions.

2.1.3 Dimensions

Dimensions contain the qualitative labels that characterize the cube's data. For example, the PRODUCT dimension contains characteristics such as PRODUCT FAMILY and PRODUCT DEPARTMENT.

A dimension is organized hierarchically; each level of the hierarchy represents the next level of granularity in the dimension. For example, **Figure 2-2** depicts the PRODUCT dimension, which has six levels. Each level except the bottom level is the parent of the level below it; you can also say that each level except the top level is the child of the level above it.

For example, PRODUCT FAMILY is the parent of the PRODUCT DEPARTMENT, which is the parent of PRODUCT CATEGORY. The most granular level is the PRODUCT NAME; it represents a specific product, such as *ADJ Rosy Sunglasses*.

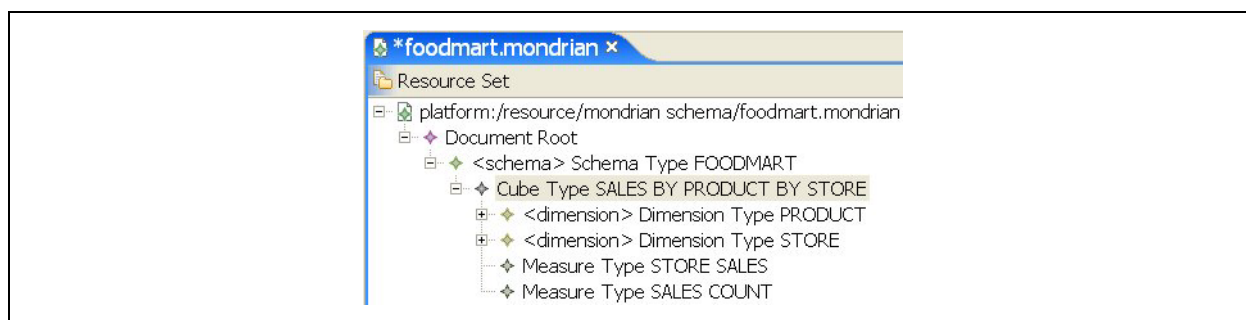


Figure 2-2 Hierarchy for the FoodMart Cube

2.1.4 Measures and Facts

Measures and facts provide the quantitative information in a cube. A measure is a formula; the value that such a formula yields is a fact.

For example, the STORE SALES measure represents the dollar amount of sales figures for a given store (or collection of stores, depending on the STORE dimension); the SALES COUNT measure represents the unit count of goods sold by a given store. These figures are aggregations of the transactional data, using summary functions such as SUM. In the example shown in [Figure 2-3](#), STORE SALES is the SUM of the store_sales column in the transactional data.

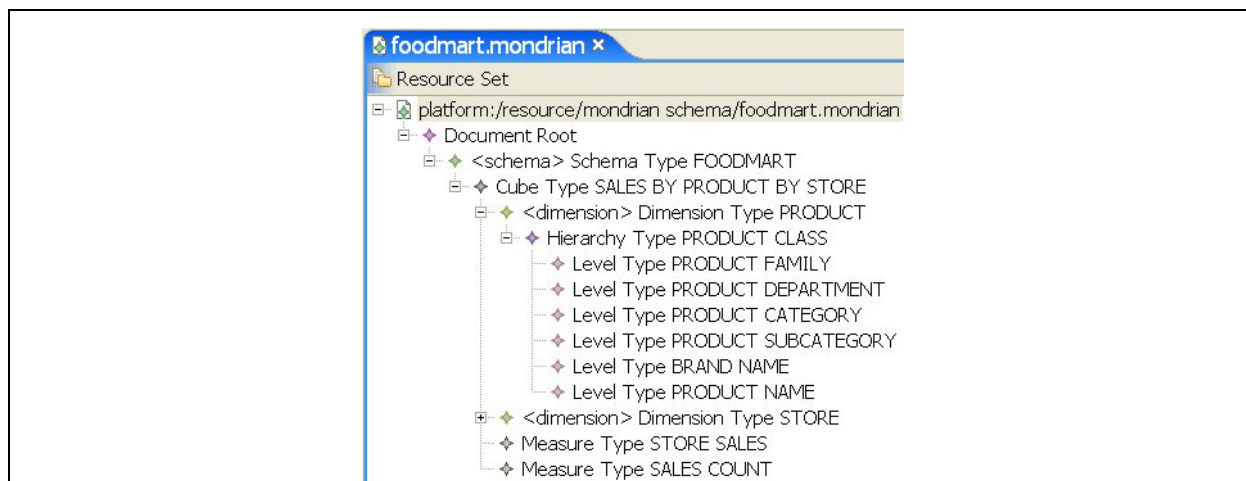


Figure 2-3 Measure Created with the SUM Aggregation Function

2.2 Jaspersoft OLAP and the Repository

Jaspersoft OLAP extends the JasperReports Server web application to provide these OLAP features:

- Browser-based OLAP interactive views called OLAP views.
- OLAP objects managed in JasperReports Server's repository, including browser-based maintenance and object-level security.
- JasperReports can use Jaspersoft OLAP connections and MDX as a basis for report data, gaining access to advanced scalability and calculations.
- XML/A services: Client applications, such as Excel, can run MDX queries against Jaspersoft OLAP cubes through the XML/A web services protocol. Jaspersoft ODBO Connect connects Excel Pivot Tables to cubes through XML/A. You can purchase ODBO Connect from Jaspersoft's store at <http://www.jaspersoft.com/jaspersoft-odbo-connect>.
- Jaspersoft OLAP commercial editions provide the following:
 - OLAP data-level security based on user profiles.
 - Browser-based management of run time parameters.

- ♦ Jaspersoft OLAP performance monitoring.
- ♦ Ad Hoc topics can be based on Jaspersoft OLAP connections.

In the JasperReports Server repository (**Figure 2-4**), Jaspersoft OLAP resources are stored in folders; security for each folder can be controlled separately. The repository is displayed in two panels. The Folders panel displays all the folders in the repository, while the Repository panel displays the contents of the selected folder.

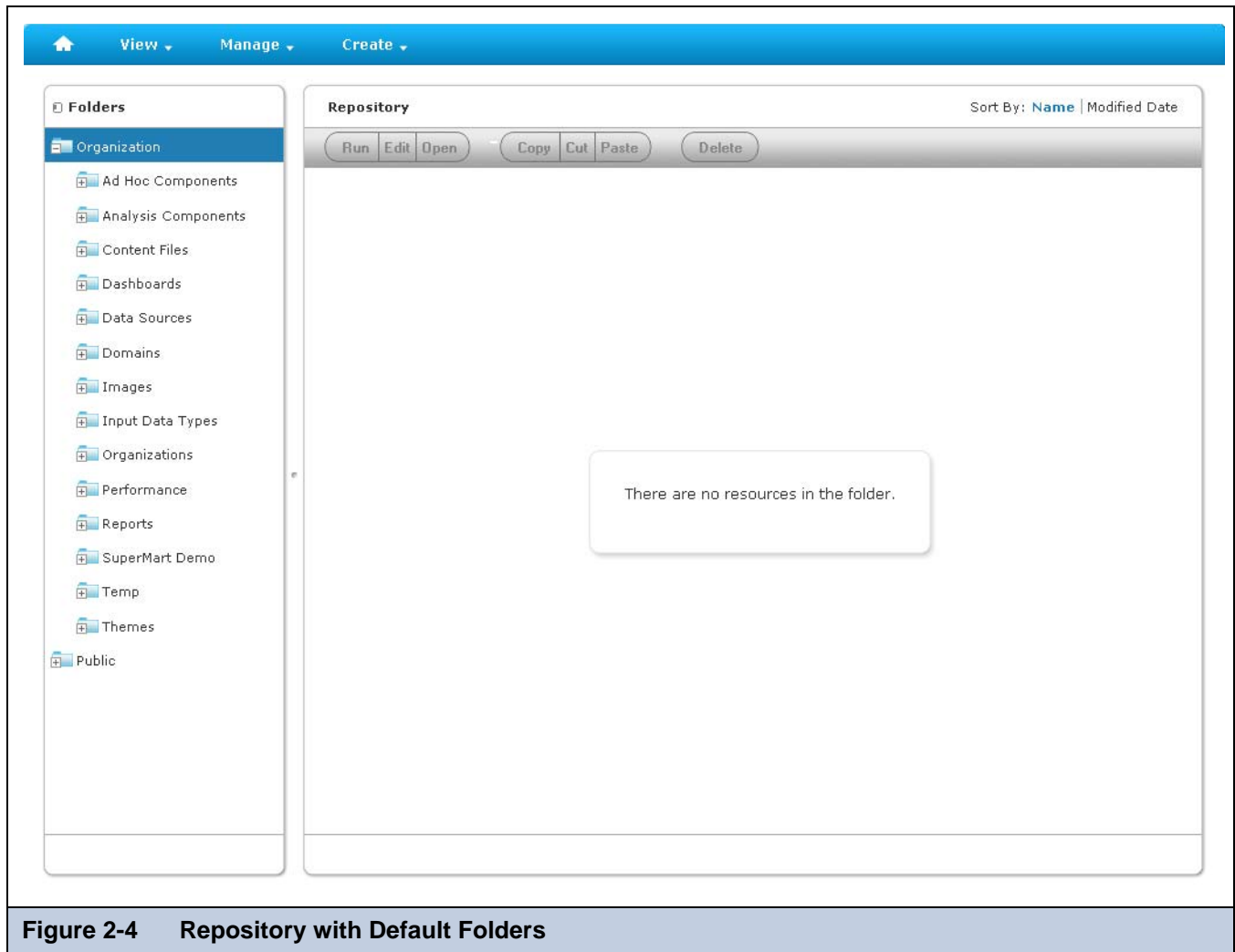


Figure 2-4 Repository with Default Folders

For more information on folders and the repository, refer to the *JasperReports Server User Guide*.

2.3 Creating a Jaspersoft OLAP Environment

This section explains how to prepare data to be used in OLAP views. These procedures can help you understand how to analyze your own data in Jaspersoft OLAP.



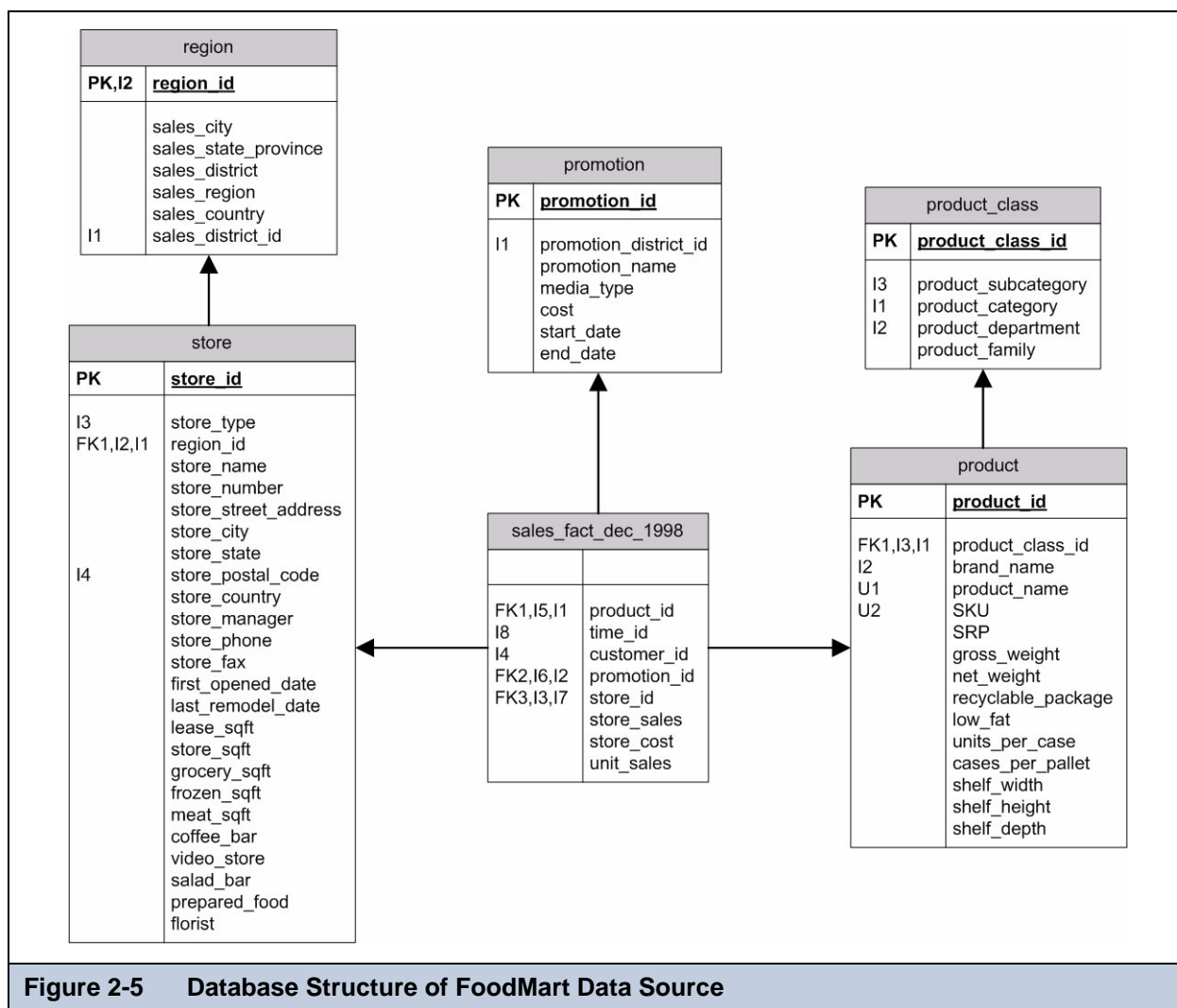
If you installed from the installer and chose the sample data option, Jaspersoft OLAP includes fully-prepared sample data, including the FoodMart sample described in this section. The installation and configuration steps in this sections are not necessary if you installed the sample data; they are instead meant to illustrate the overall process of implementing an OLAP view.

2.3.1 Creating the Database

Jaspersoft OLAP uses data from a SQL database, and uses a Relational OLAP (ROLAP) architecture. Unlike multi-dimensional (MOLAP) tools, such as Hyperion and Cognos, Jaspersoft OLAP doesn't process data from the database into an intermediate form. Typically, it runs against star and snowflake schemas, which are database structures used for analysis.

From a business perspective, consider the case of a retail business that has gathered data regarding point-of-sale transactions as well as information on product and store locations. These data are transformed into a multidimensional database that is de-normalized to optimize for retrieval and for OLAP operations. An example of a multidimensional database structure is shown in [Figure 2-5](#). A more complete example is shown in [A.1, "FoodMart Database Schema," on page 102](#).

This example assumes you are using Relational On-line Analytical Processing (ROLAP).



The sample data source is organized in a data structure known as the star-schema, where the measures are stored in the fact tables, and the dimensions are stored in the dimension tables. For example, the SALES_FACT_DEC_1998 is a fact table containing measures, such as STORE_SALES, STORE_COST and UNIT_SALES. The foreign keys STORE_ID, PROMOTION_ID and PRODUCT_ID are associated with the dimension tables STORE, PROMOTION and PRODUCT, respectively. The STORE and PRODUCT tables are further normalized by their summary tables, REGION and PRODUCT_CLASS, respectively.

The foodmart database depicted in [Figure 2-5](#) is included with Jaspersoft OLAP as both sample data that can be installed with the product and as a MySQL database script.

Using the commands in the following code sample, you can create a database in Windows at the MySQL command line:

```
C:\>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection ID is 1 to server version: 5.0.19-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database foodmart;
Query OK, 1 row affected (0.00 sec)

mysql> quit
Bye
```

2.3.2 Importing Data using an SQL Database Script

Your environment needs processes that set up the data for the cubes and dimensions and populate the database with regular updates. These processes can be created by Jaspersoft ETL, a graphical tool and runtime environment that allows you to create data transformation processes. Other data integration tools and hand-scripting can also be used. In general, the data integration work to create and maintain the cube databases consumes 50-90% of the effort of setting up an OLAP environment.

For our example, we run a MySQL database script to populate our database; the following script shows the steps to populate a FoodMart database.

```
C:\>mysql -u root -p -D foodmart < FoodMart-MySQL.sql
Enter password: *****

C:\>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection ID is 3 to server version: 5.0.19-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+-----+
| Database          |
+-----+
| information_schema |
| foodmart           |
| jasperserver       |
| mysql              |
| test               |
+-----+
9 rows in set (0.27 sec)

mysql> quit
Bye
```

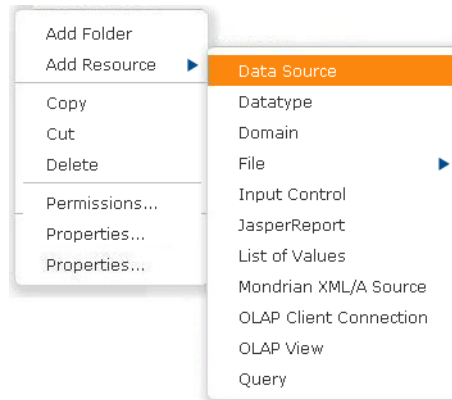

With your data populated, you can now create a data source in JasperReports Server that points to this data.

2.3.3 Defining a Data Source in JasperReports Server

Now that we have a database, let's define a data source that points to the MySQL catalog (that is, the database).

To create a Foodmart data source:

1. Click **View > Repository** to open the repository.
2. In the Folders panel, right-click navigate to Organization/Analysis Components, right click the Analysis Data Sources folder, and select **Add Resource > Data Source** in the context menu:



The Add Data Source panel appears with the Set Data Source Type and Properties dialog.

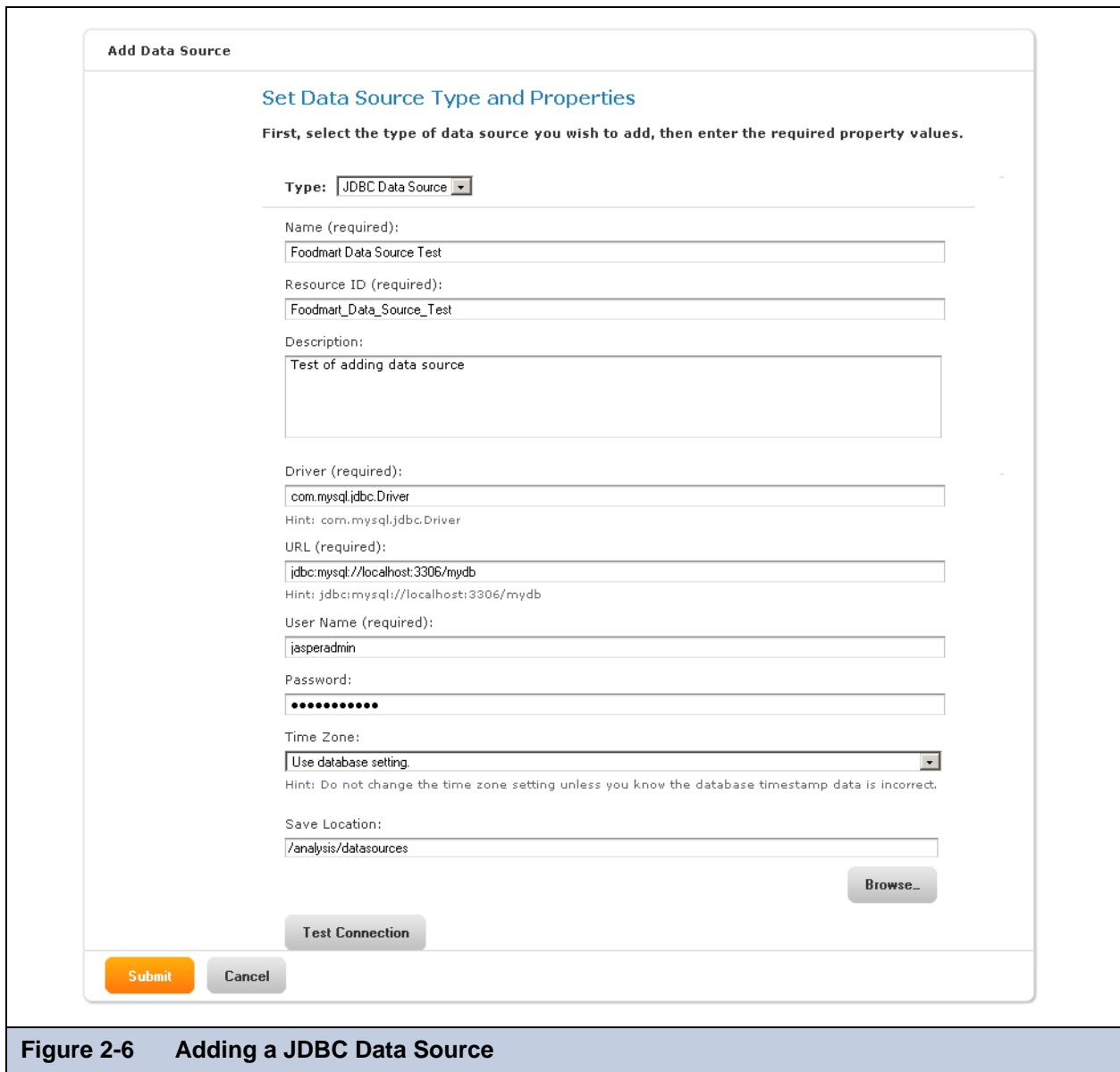


Figure 2-6 Adding a JDBC Data Source

3. In the Type drop-down, select **JDBC Data Source**.
4. Enter identifying data for the data source:
 - **Name:** Foodmart Data Source Test
 - **Description:** Test of adding data source
 - **Driver** (for MySQL): com.mysql.jdbc.Driver
 - **URL** (for MySQL): jdbc:mysql://localhost:3306/foodmart

The resource ID is entered automatically. You can change it as needed.

In the URL, 3306 is the MySQL database port and foodmart is the MySQL database name.

5. Enter your user name and password. Only enter a time zone if your database has an incorrect timestamp.
6. Click **Submit**.

The new JDBC data source, Foodmart Data Source Test, appears in the repository.

2.3.4 Creating and Uploading an OLAP Schema

In the context of JasperReports Server, an OLAP schema is a file resource stored in the repository. In a broader context, the OLAP schema defines a multidimensional structure containing cubes, dimensions, and measures. The cubes represent a real-

world entity (such as annual sales), where the dimensions are the characteristics of the cube (such as product and store), and measures are the quantitative information (such as store sales and sales count).

The figure below shows the FoodMart OLAP schema that is distributed with JasperReports Server. The schema describes how the tables in a database structure can be viewed as a cube.

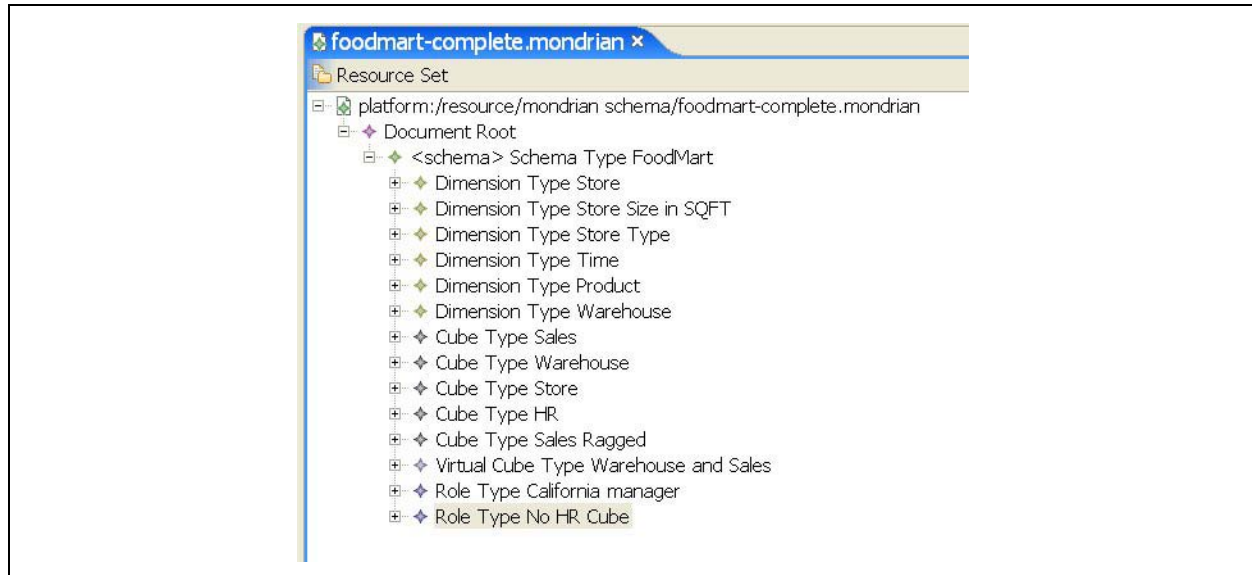
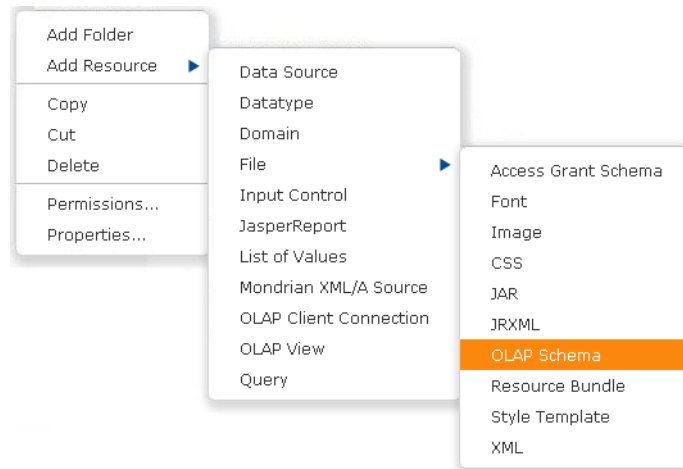


Figure 2-7 FoodMart OLAP Schema

Let's add this schema to the repository.

To add an OLAP schema:

1. Click **View > Repository** to open the repository.
2. In the Folders pane, right-click the Organization/Analysis Components/Analysis Schemas folder and select **Add Resource > File > OLAP Schema** in the context menu:



The Add File panel appears with the Upload a File from Your Computer dialog.

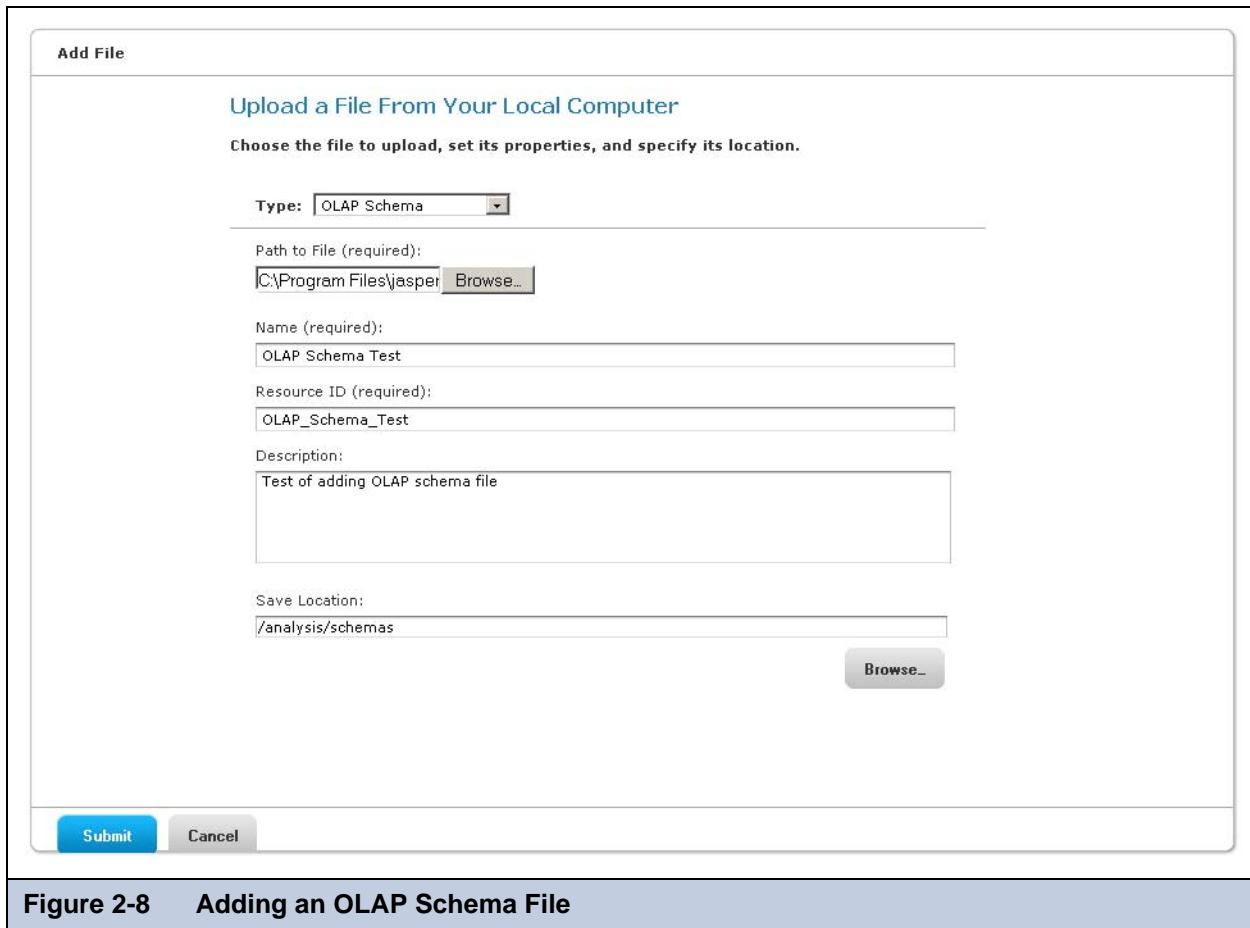


Figure 2-8 Adding an OLAP Schema File

3. In the **Type** drop-down, select OLAP Schema.
4. Click **Browse** to browse to an OLAP schema file that is in your local file system.
A sample schema file for FoodMart can be found in the <js-install/samples/schemas> directory of the JasperReports Server installation.
5. Enter a name and description for the file:
Name: OLAP Schema Test
Description: Test of adding schema file
The resource ID is created automatically from the name; you can change it as needed.
6. In the **Save Location** field, enter the path to the folder where the schema should be saved.
The default location is /analysis/schemas. Enter a different path if schema files are stored elsewhere in your system.
7. Click **Submit**.
The new schema, OLAP Schema Test, appears in the repository in the Organization/Analysis Components/Analysis Schemas folder.

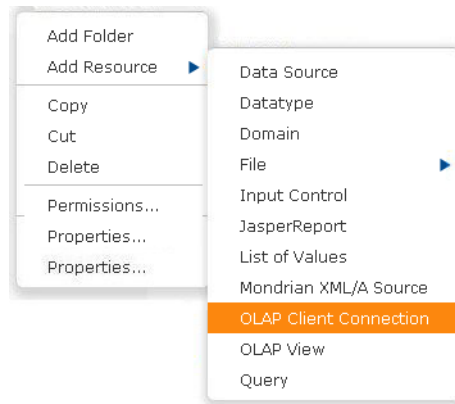
2.3.5 Creating a Mondrian Connection

An OLAP client connection defines the type of connection and the source of the data. There are two types of OLAP client connections: Mondrian connections refer to local data sources in the repository; XML/A connections refer to data sources stored in a different repository or system. Only Mondrian connections can directly include an access grant definition. For more information on XML/A, refer to section 5.2.1, “XML/A Definitions,” on page 85.

To create a Mondrian connection:

1. Click **View > Repository**.

- In the Folders pane, right-click the Organization/Analysis Components/Analysis Connections folder.
- In the menu that appears, click **Add Resource > OLAP Client Connection**:



The Add OLAP Client Connection panel appears with the Set Connection Type and Properties dialog.

 A screenshot of a dialog box titled 'Add OLAP Client Connection'. The dialog has a header 'Set Connection Type and Properties' and a sub-header 'First select the type of connection you want to add, then enter the required property values.' Below this, there is a 'Type' dropdown menu set to 'Mondrian'. Underneath are three input fields: 'Name (required):' with the text 'Test Mondrian Connection', 'Resource ID (required):' with the text 'Test_Mondrian_Connection', and 'Description:' with the text 'Test of Adding Mondrian connection'. At the bottom, there is a text field containing '/analysis/connections' and a 'Browse...' button. At the very bottom of the dialog are three buttons: 'Previous', 'Next', and 'Cancel'.

Figure 2-9 Setting the Connection Type and Properties

- In the **Type** drop-down, click **Mondrian**.
- Enter a name and description for the new connection. For instance, use the following:
Name: Test Mondrian Connection
Description: Test of adding Mondrian connection
 The resource ID is created automatically from the name; you can change it as needed.
- Click **Next**.

The Locate OLAP Schema dialog appears.

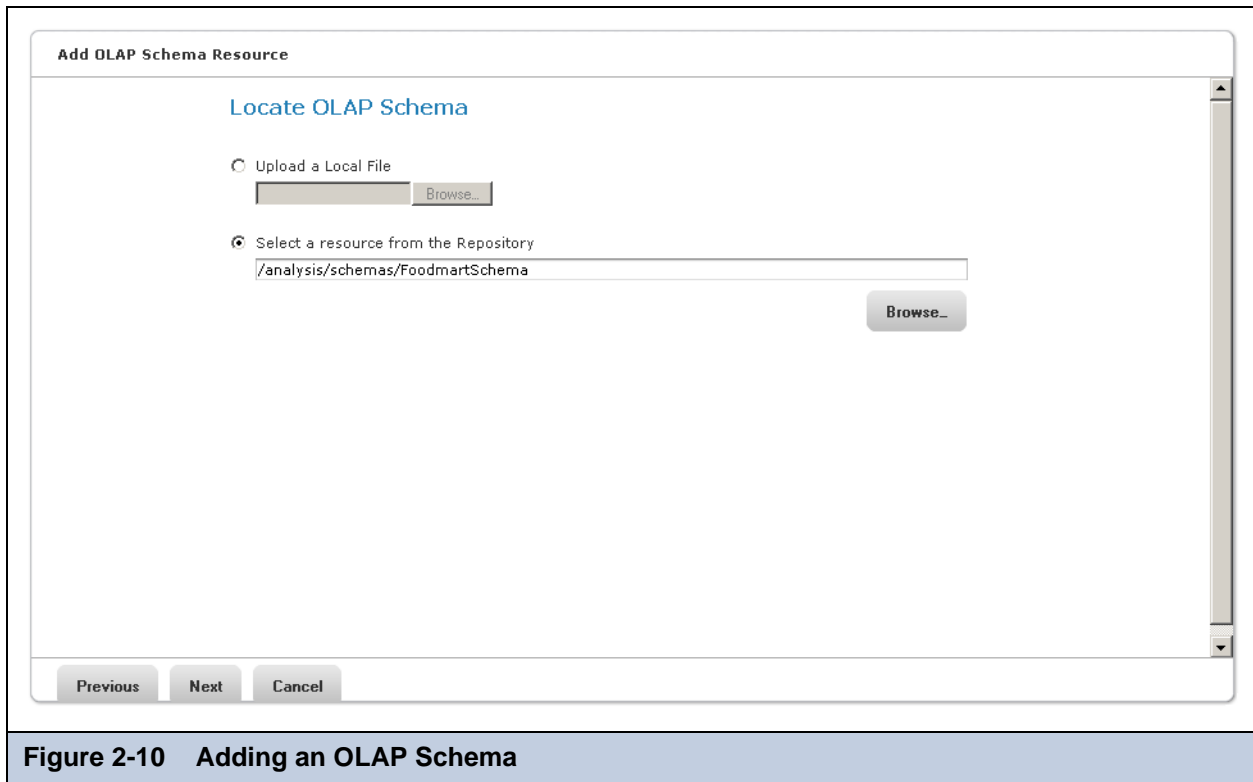


Figure 2-10 Adding an OLAP Schema

7. You can select your a local schema or a default schema from the repository.
In this example, we select the default Foodmart schema:
 - a. Click **Select a resource from the Repository** then click **Browse**.
The Select Resource from Repository dialog appears.

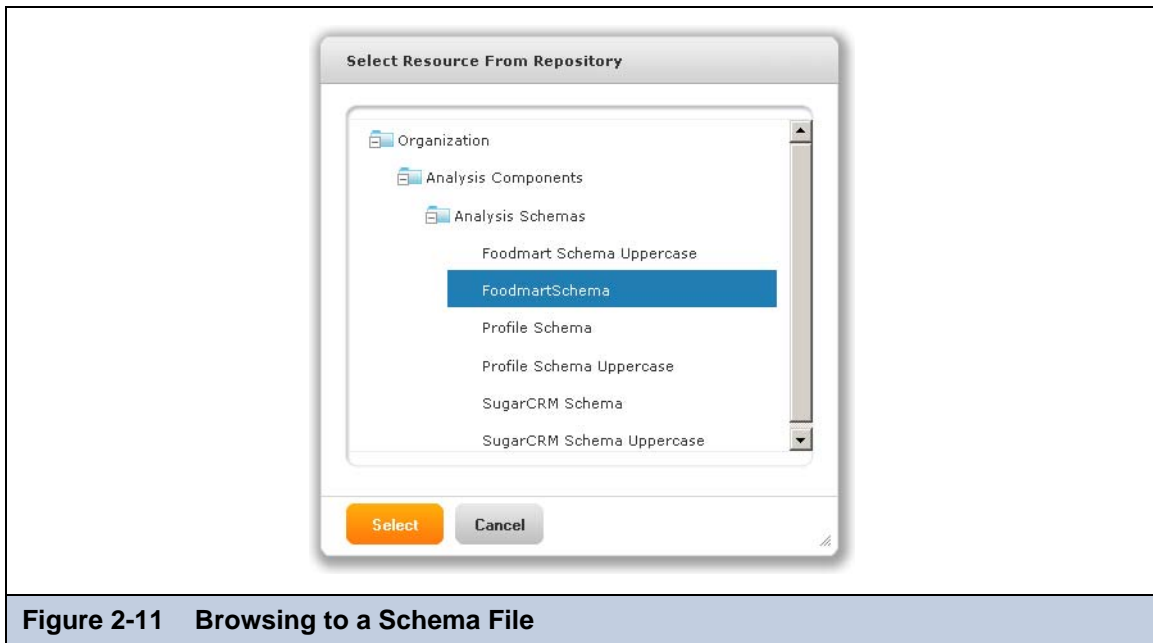


Figure 2-11 Browsing to a Schema File

- b. Use the + buttons on the folders (do not double-click the folders) to expand the folders to the Organization/Analysis Components/Analysis Schemas folder.
- c. In the folder, select **FoodmartSchema** then click **Select**.

The Locate OLAP Schema dialog appears again with the selected schema.

8. Click **Next**.

The OLAP Schema Resource dialog appears. The fields already contain the data that you entered.

OLAP Schema Details

OLAP Schema Resource

Set the properties for the resource.

Type: OLAP Schema

Selected Resource:
/analysis/schemas/FoodmartSchema

Name (required):
FoodmartSchema

Resource ID (required):
FoodmartSchema

Description:
Foodmart Analysis Schema

Save Location:
/analysis/schemas

Browse...

Previous Next Cancel

Figure 2-12 Locating a Data Source

9. Verify that the data is correct, then click **Next**.

The Locate Data Source dialog appears.

10. To select a data source, click **Select a Data Source from the repository** and browse (as in [step 7](#)) to Organization/Analysis Components/Data Sources/JServer Jdbc Data Source.

Add Data Source

Locate Data Source

Define a Data Source in the next step

Select a Data Source from the repository

/datasources/JServerJdbcDS

Browse...

Previous Next Cancel

Figure 2-13 Browsing to Data Source

11. Click **Next**.

The Locate Data Source dialog appears again with the data source information that you entered.

12. Click **Next**.

The Locate Access Grant Definition panel appears.

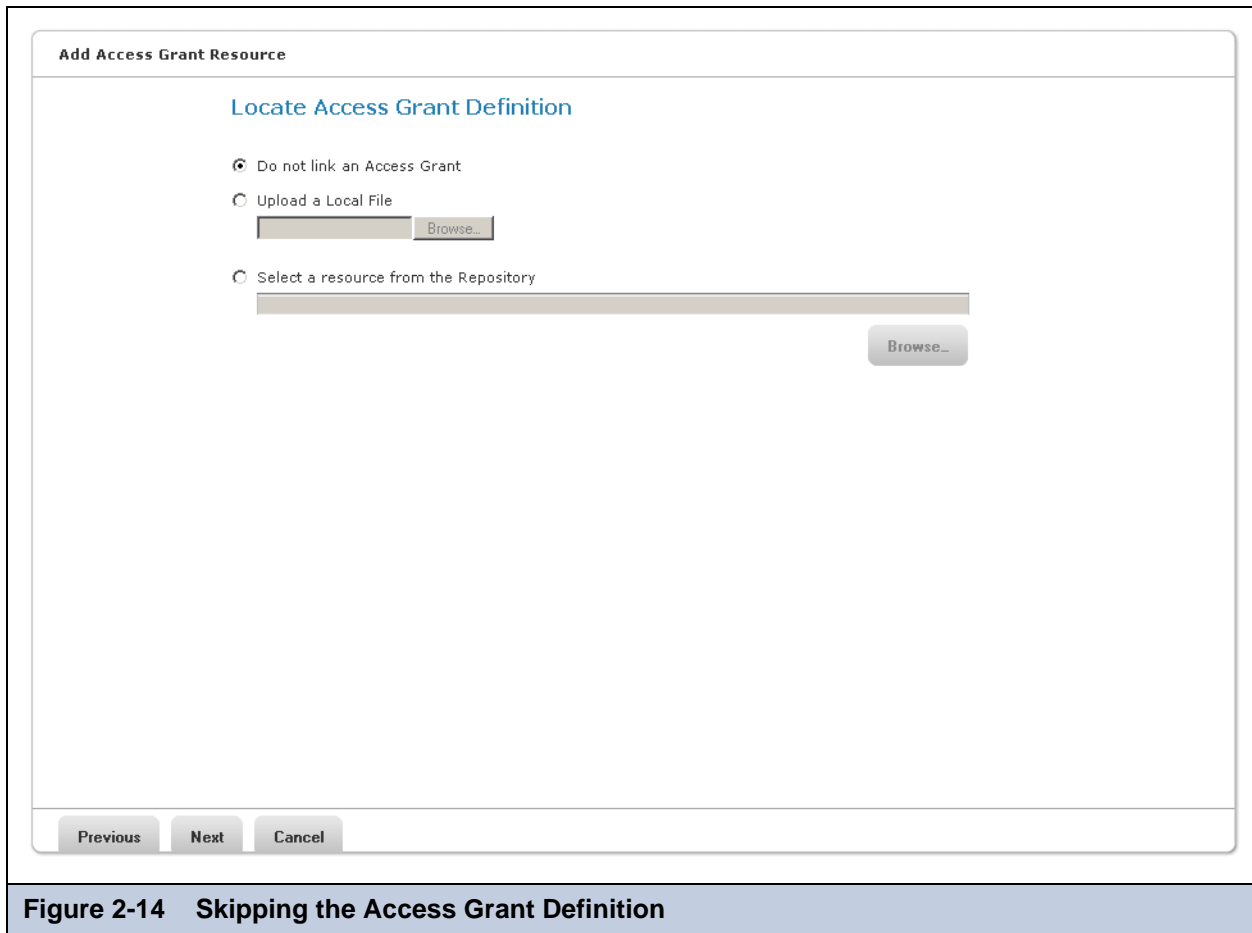


Figure 2-14 Skipping the Access Grant Definition

13. Select **Do not link an Access Grant** and click **Next**.

You can select an access grant definition from the repository, upload one from the file system, or skip the step. In this example, we skip the step.

The repository appears. A confirmation message appears at the top of the window and the new Mondrian connection (Test Mondrian Connection) appears in the repository



If you clicked **Define a Data Source in the next step** in [step 11](#) instead of **Next**, you would proceed through a series of windows to define the data source as JDBC, Bean, or JNDI, and you would supply the appropriate connection information. For more information about defining data sources, refer to the *JasperReports Server Administrator Guide*.

2.3.6 Defining an MDX Query

Each OLAP view operates against a particular cube within the OLAP connection. The view also includes an MDX query that defines the data to display.

MDX is a query language used to define the contents of an OLAP view. Originating at Microsoft, the language is widely used for multidimensional data retrieval. An MDX query, coupled with an OLAP schema, retrieves data from a database. The following is a typical MDX query:

```
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]}
ON COLUMNS, {[Promotion Media].[All Media], [Product].[All Products]} ON ROWS
from [Sales]
where [Time].[2006].[Q4].[12]
```

In the example MDX:

- [Measures].[STORE SALES] is a measure.
- [PRODUCT].[ALL PRODUCTS] is a member in the PRODUCT dimension.
- [STORE].[ALL STORES] is a member in the STORE dimension.
- [SALES BY PRODUCT BY STORE] is a cube.
- [TIME].[2006] is a member used as a filter axis in the WHERE clause.

Jaspersoft OLAP constructs MDX queries as you use its controls to manipulate the view. If you aren't proficient with MDX, you may want your initial view to use the simplest possible MDX query. For example, you can start with one measure and one dimension, then use the tool bar buttons and navigation table to tailor the view to show the exact data you want. You can always see the MDX that underlies the current view by clicking **View MDX query** in the tool bar. You can also edit this query to change the view if you are comfortable with MDX.

2.3.7 Configuring XML/A

You can expose your Mondrian OLAP Connections to XML/A web services clients by configuring Mondrian XML/A Definitions. OLAP views can use XML/A to run against such remote servers, which might be based on applications other than Jaspersoft OLAP. Jaspersoft ODBO Connect can leverage XML/A definitions to provide OLAP slice and dice in Excel Pivot Tables. If you use the Community Project of Jaspersoft OLAP, you can purchase ODBO Connect from Jaspersoft's store at <http://www.jaspersoft.com/jaspersoft-odbo-connect>. If you use a commercial edition, you can download this application from the Support Portal.

For more information on XML/A configuration, refer to **5.2.1, "XML/A Definitions," on page 85**.

CHAPTER 3 ANALYZING DATA IN A VIEW

This chapter explains how to analyze data using OLAP views. It has these sections:

- **OLAP Views**
- **Analyzing Data in an OLAP View**
- **Using the OLAP View Tools**
- **Troubleshooting Jaspersoft OLAP**

The following sections assume you have installed the sample data or followed the installation instructions in [2.3, “Creating a Jaspersoft OLAP Environment,”](#) on page 14.



Some users analyze their data in Excel using Jaspersoft ODBO Connect. For information on this feature, refer to the Jaspersoft ODBO Connect documentation.

3.1 OLAP Views

An OLAP view (also called an analysis view) represents a selective set of multidimensional data in a given cube. It is the entry point to Jaspersoft OLAP analytics operations. It is usually defined to meet a specific need, such as to analyze sales trends or profile customer demographics.

The view is defined in terms of an OLAP connection and an MDX query. The OLAP connection specifies the data access properties, while the MDX query determines the contents of the view. You can create a view with a Mondrian connection or an XML/A connection.

To open the sample OLAP view:

1. Click **View > OLAP Views**.
A list of OLAP views appears in the Repository panel.
2. Click the name **Foodmart Sample OLAP View** to open the view.
The view displays the tool bar and navigation table.

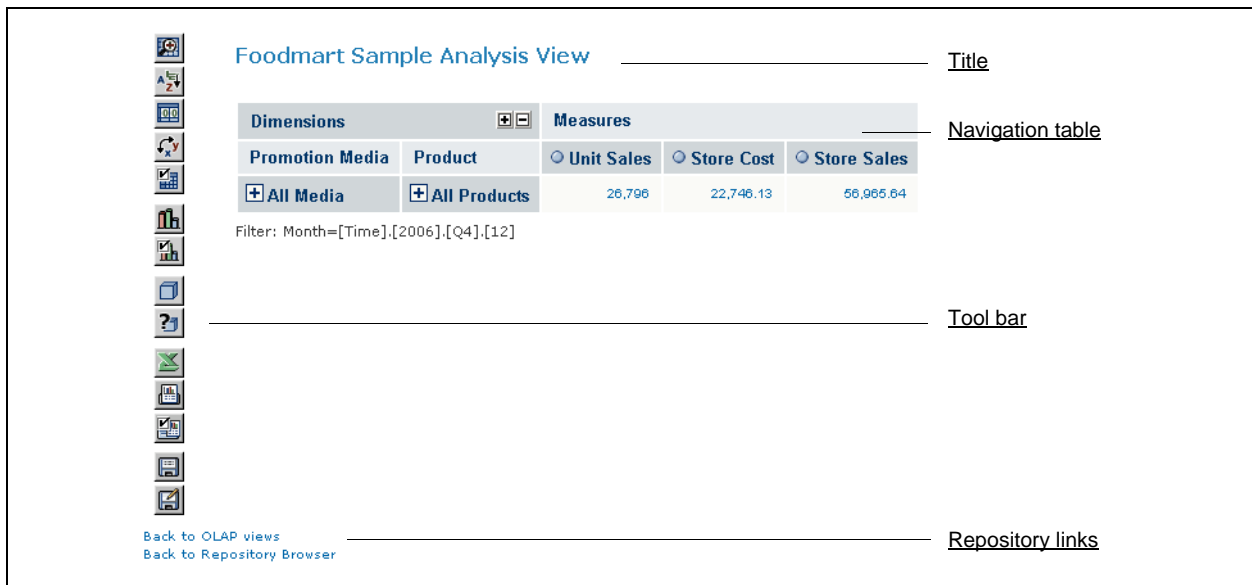


Figure 3-1 Mondrian OLAP View

To create an OLAP view:



The following example creates an OLAP view using a Mondrian connection. XML/A connections can be used, as well.

1. Click **View > Repository** to open the repository.
2. In the Folders pane, right-click the Analysis Components/Analysis Views folder.
3. In the menu that appears, click **Add Resource > OLAP View**.

The Add OLAP View panel appears with the Name the View dialog.

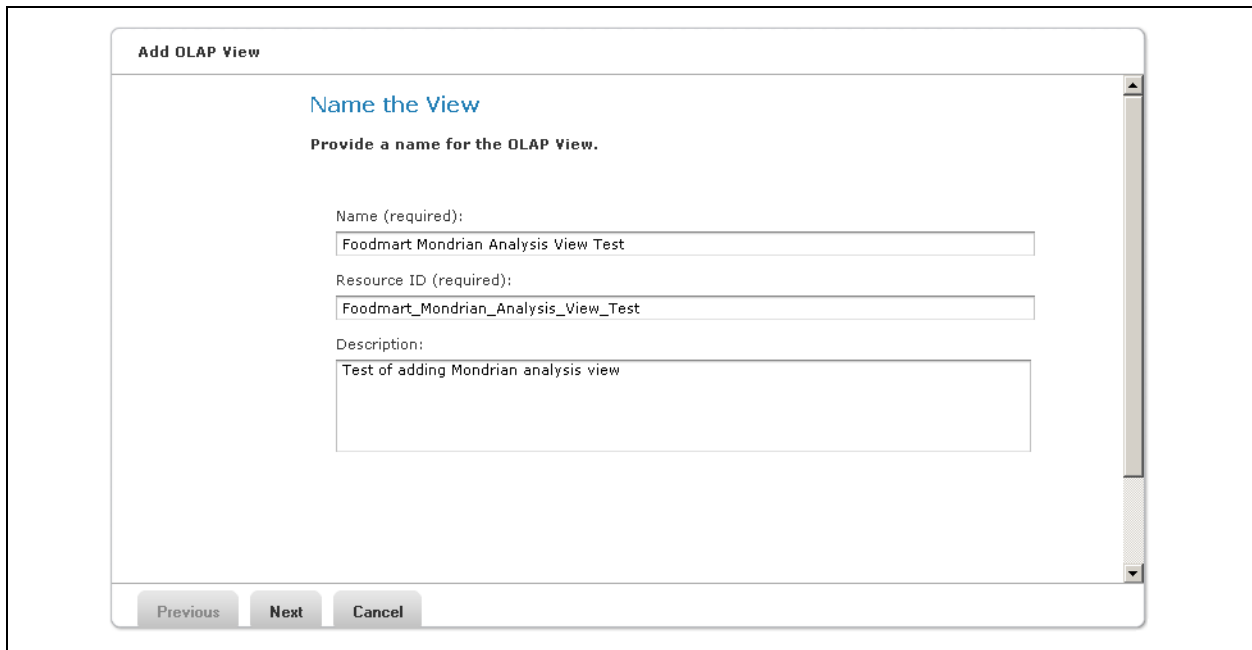


Figure 3-2 Naming the New OLAP View

4. Enter a name and description for the view:
 - ♦ **Name:** Mondrian OLAP View Test
 - ♦ **Description:** Test of adding Mondrian OLAP view for the description.

The resource ID is created automatically from the name; you can change it if necessary.

5. Click **Next**.

The Add Mondrian OLAP Client Connection Source panel appears with the Locate Mondrian Client Connection Source dialog.

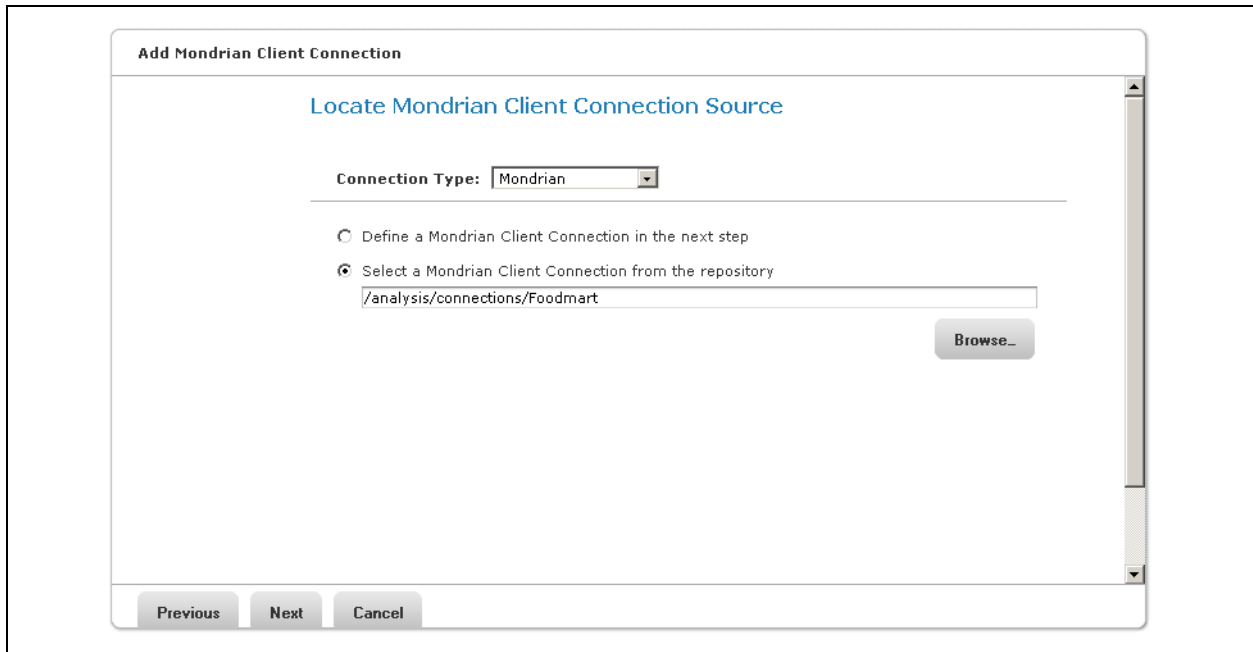


Figure 3-3 Selecting a Mondrian OLAP Client Connection

6. In the **Connection Type** drop-down, select Mondrian.
7. Select the **Select a Mondrian Client Connection from the repository** option, then browse to Organization/Analysis Components/Analysis Connections/Foodmart.
If you want to create a new connection instead, use the same procedure as in [step 7 in section 2.3.5, "Creating a Mondrian Connection," on page 22](#).
8. Click **Next**.

The Add Query panel appears with the Define the Query dialog.

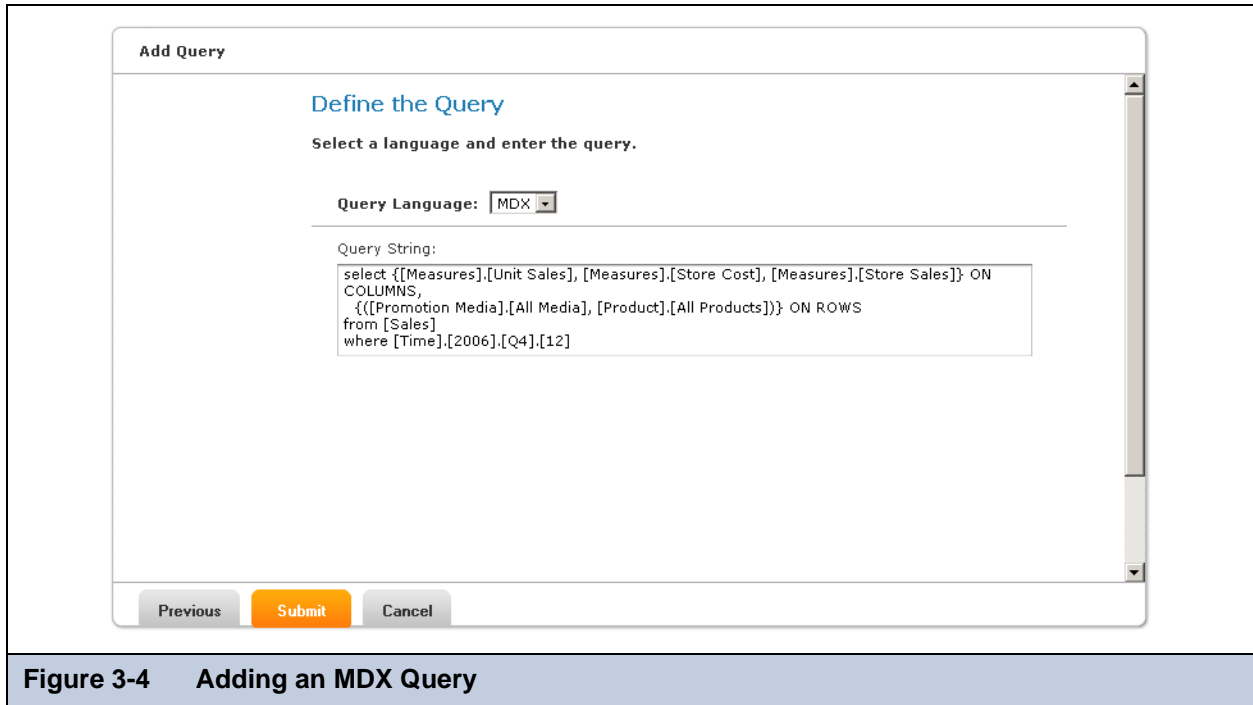


Figure 3-4 Adding an MDX Query

9. In the **Query Language** drop-down, select **MDX**.
10. Enter a valid MDX query, such as the one shown in 2.3.6, “**Defining an MDX Query,**” on page 24.
For help writing MDX queries, see that section as well as 3.2, “**Analyzing Data in an OLAP View,**” on page 30.
11. Click **Submit**.
The new view appears in the Repository panel.

3.2 Analyzing Data in an OLAP View

The purpose of data analysis is to uncover relationships and trends in the data. The analysis should give you new insights into the situation that the data describes. To structure the analysis, you should ask questions like these:

- How did my organization perform this year as compared to last year? Which parts and personnel of the organization did better and which did not?
- For a consumer business, what is my most and least profitable product/customer/salesperson/office/store? Which factors in my data trend in the same direction as the most profitable, which factors trend in the opposite direction, and which are neutral? How do those factors trend for the least profitable?
- For a hospital, for instance, which patients are staying longer than is typical for their diagnosis? What symptoms and other diagnoses do the long-staying patients have?

The questions you can answer with Jaspersoft OLAP depend on:

- The available data.
- The structure of the data in terms of OLAP; that is, the structure of your cubes, dimensions, and measures.
- The starting OLAP view defines which cube you want to analyze and the metrics relevant to a particular need.

OLAP views give business users a starting point for analysis that can then be sliced and diced to answer detailed questions. For a particular OLAP data set, there are usually a number of OLAP views defined as convenient entry points.

As an example, let's answer a specific question: “What is the quarterly sales dollar amount for the snack foods category in 2006 for stores in California?”

3.2.1 Displaying Product Sales by Quarter

Jaspersoft OLAP is well suited to comparing numeric data across a period of time or by product category. That's how we'll answer our example question.

To open the sample OLAP view:

1. Click **View > OLAP Views**.
A list of OLAP views in the repository appears.
2. Click **Foodmart Sample OLAP View** to open it.
The default navigation table appears along with the Jaspersoft OLAP tool bar.

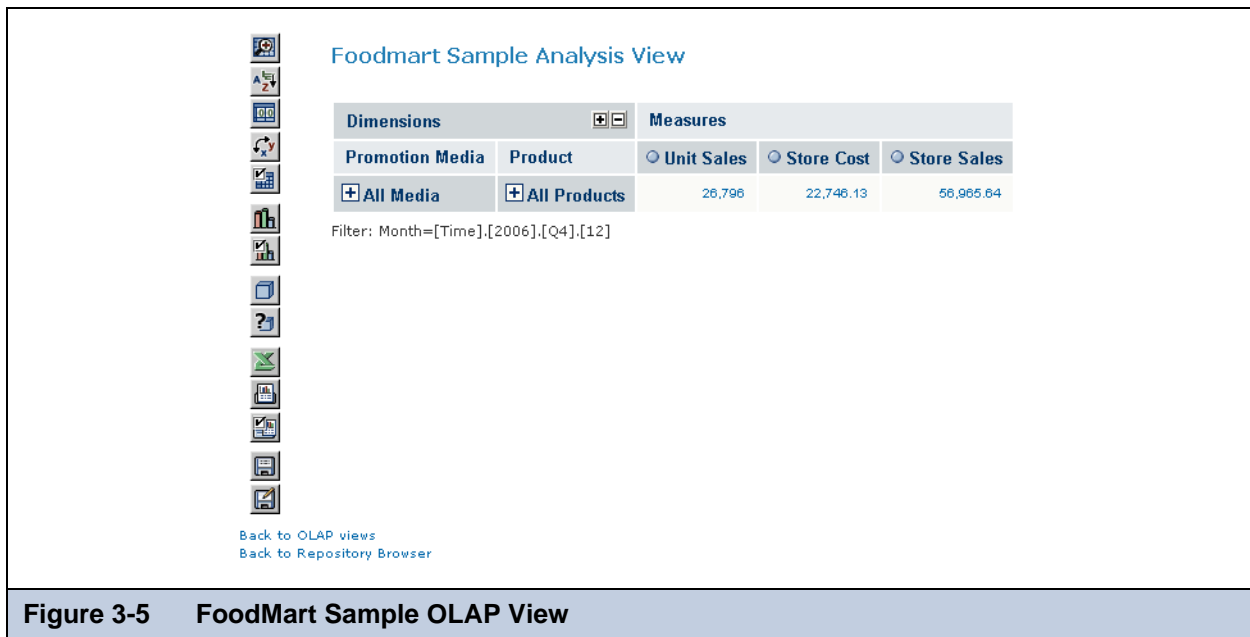



Figure 3-5 FoodMart Sample OLAP View

To find the quarterly sales dollar amount for the snack foods category in 2006 for stores in California:

1. Click the **Change Data Cube** tool bar button: .
The Change Data Cube dialog appears.

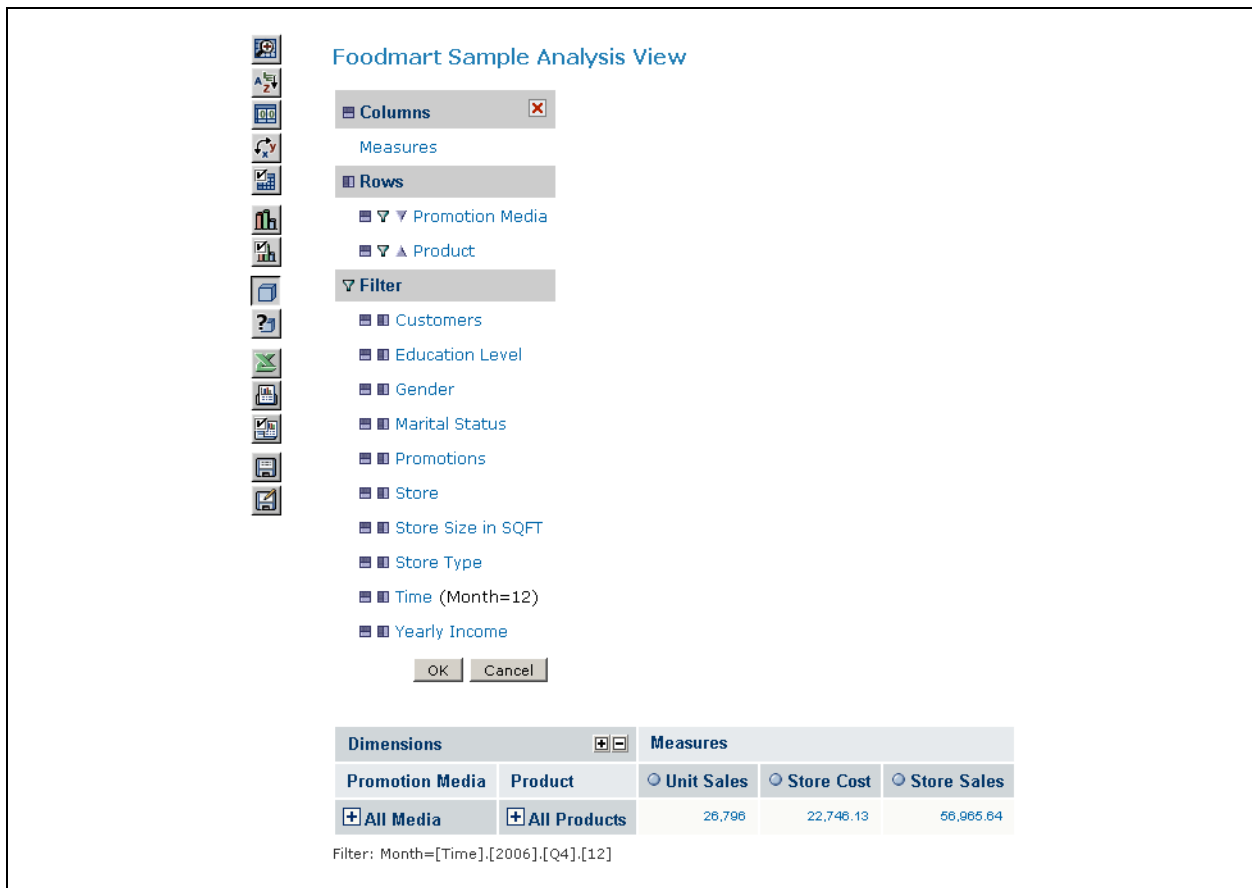



Figure 3-6 Change Data Cube

- Click the **Move to Columns** icon  next to the Time filter to make Time a column. Note that the filter is set to Month=12.

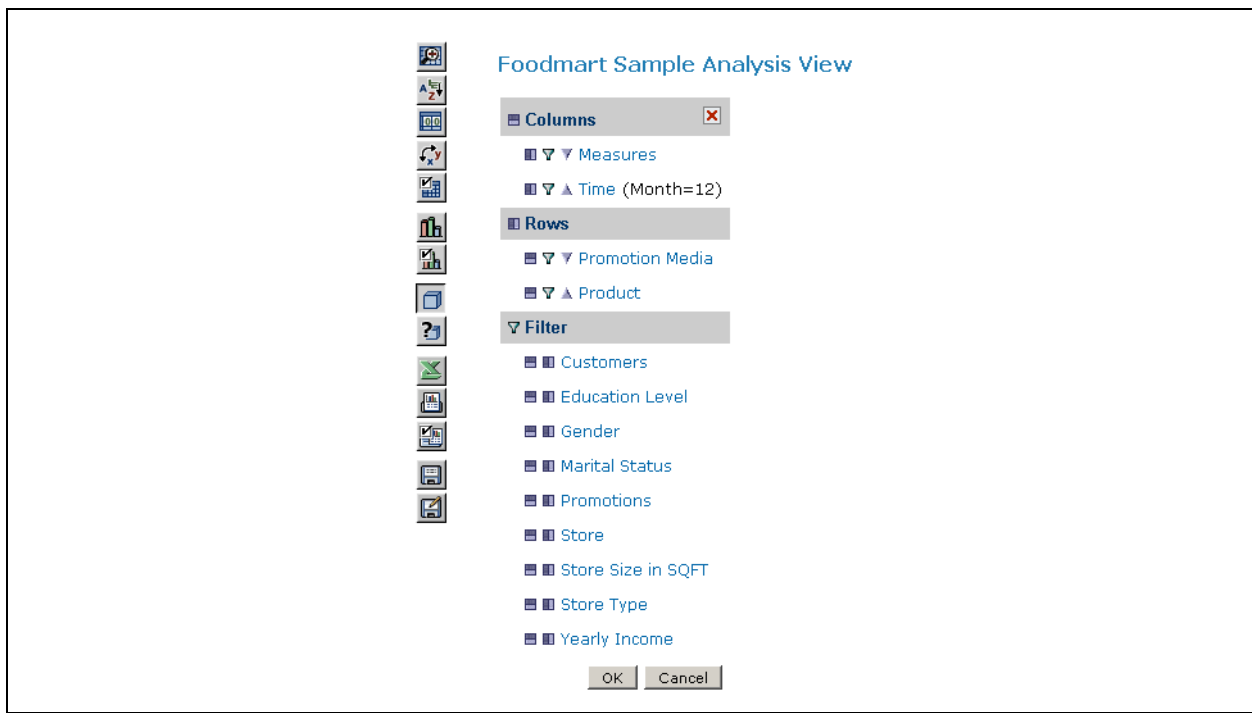


Figure 3-7 Adding a Time Column

3. Click **Time** to open a tree displaying the dimension members.
4. Expand Time by clicking **+** next to it, and select **2006**.

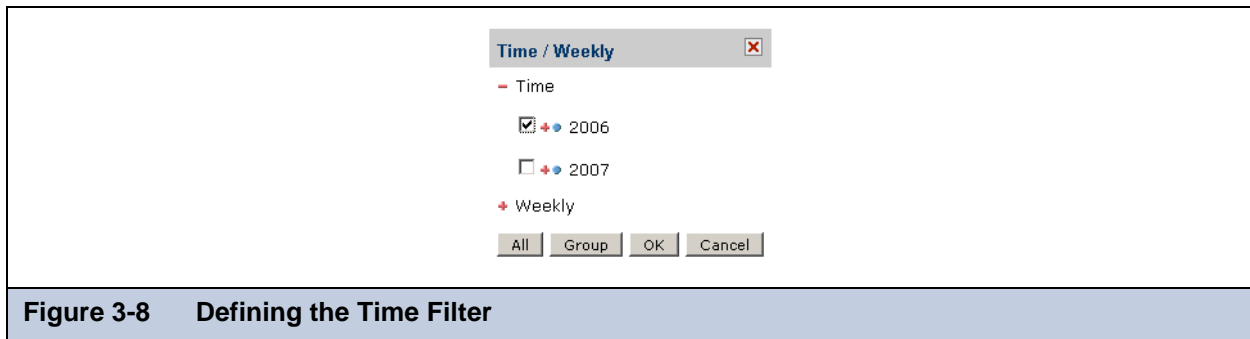


Figure 3-8 Defining the Time Filter

5. Click **OK** to close the tree and return to the Change Data Cube dialog.



Changes you make in the Change Data Cube dialog aren't applied to the view until you click **OK** again to close the dialog. Thus, while the Change Data Cube dialog is open, the view won't match your selections in the dialog.

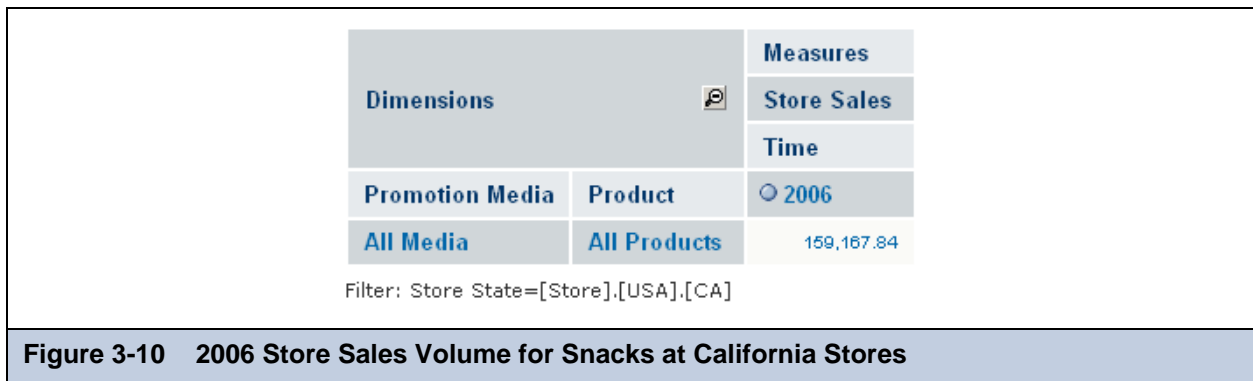
6. In the Change Data Cube dialog, click **Store** in the Filter list to change the filter.
The Store filter appears.
7. Expand **All Stores** by clicking **+** next to it, then expand **USA**, and select **CA**.

Dimensions		Measures		
Promotion Media	Product	Unit Sales	Store Cost	Store Sales
+ All Media	+ All Products	26,796	22,746.13	56,965.64



Filter: Month=[Time].[2006].[Q4].[12]


Figure 3-9 Defining the Store Filter

8. Click **OK** to close the tree and return to the Change Data Cube dialog.
9. Click **Measures** and clear the check boxes next to Unit Sales and Store Cost to remove them.
10. Click **OK** twice to accept the selections.
Jaspersoft OLAP updates the view to match your selections.



11. Ensure that Zoom on Drill is active by checking whether its icon is pressed:

- When **Zoom on Drill** is active, its tool bar button looks like this .
- When **Zoom on Drill** isn't active, its tool bar button looks like this . Click it to enable zooming when you click a dimension member.

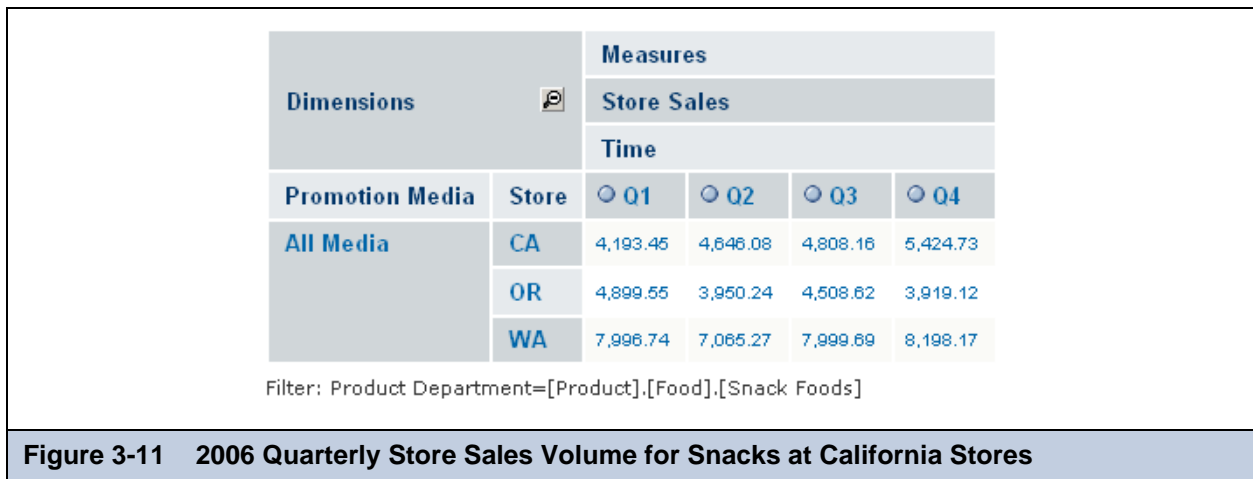
Now, when the cursor hovers over a dimension member, it changes to .

12. Click **2006** to zoom into 2006 data.

13. Click **ALL PRODUCTS**, **Food**, and **Snack Foods** to zoom into this data.

14. Click the **Edit Display Options** tool bar button, click the **Show all parent columns** check box to clear it, and click **OK**.

The following navigation table appears:




This view now shows the total sales for snack food in stores in California for 2006, broken down by quarter.

The following sections build on this example.

3.2.2 Displaying Product Sales by Time across Store Locations

We can continue and analyze multiple numeric values across time. Suppose we want to compare the quarterly snack foods sales dollar amount among the West Coast states.

To compare quarterly snack foods sales dollar amount among the West Coast states:

1. Click . For details about using cube dimensions, see [3.3.2.1, “Columns, Rows, and Filters,” on page 40](#).
2. Make **STORE** a row and select **USA**. Click **OK**.

3. Make **PRODUCT** a filter and select **Snack Foods**.
4. Click **OK** twice to accept the selections.
5. Click **USA** to zoom in.

The following navigation table appears.

Dimensions		Measures			
		Store Sales			
		Time			
Promotion Media	Store	Q1	Q2	Q3	Q4
All Media	CA	4,193.45	4,646.08	4,808.16	5,424.73
	OR	4,899.55	3,950.24	4,508.62	3,919.12
	WA	7,996.74	7,065.27	7,999.69	8,198.17

Filter: Product Department=[Product].[Food].[Snack Foods]







Figure 3-12 2006 Quarterly Snack Food Sales, Dollar Amount, for West Coast States

The view now shows the quarterly snack food sales compared by state.

3.2.3 Sorting the Display

Sorting allows you to display an ordered view of data. Suppose we want to list the top stores for highest sales dollar amount in the West Coast.

To sort the display of West Coast sales:

1. Click the navigation table's **Zoom Out All** button: .
2. Zoom into USA.
3. Click .
4. Move **PROMOTION MEDIA** to the filter area to remove it from the view.
5. Click **OK** twice to accept the selections.
6. Clear **Zoom on Drill** by clicking .
7. Click the navigation table's **Expand All Members**  button.
8. Select **Show all parent columns** in the Edit Display Options pane.
9. Select **Sort across cube hierarchy** and **Start sorting in descending order** in the Edit Display Options pane.
10. Click the navigation table's **Expand All Members**  button again.
11. Click the **Sort** icon  next to 2006.

The navigation table displays the top stores for the year ordered by dollar sales amount.

Dimensions					Measures							
					Store Sales							
					Time							
					2006	Q1			Q2		Q2	
Store	Store Country	Store State	Store City	Store Name		Q1	Q1	Q1	Q2	Q2	Q2	
						1	2	3	4	5		
All Stores	USA	WA			31,259.87	7,996.74	2,450.30	2,651.56	2,894.88	7,065.27	2,201.68	2,241.36
		CA			19,072.42	4,193.45	1,319.06	1,426.41	1,447.98	4,646.08	1,522.81	1,492.15
		OR			17,277.53	4,899.55	1,720.89	1,230.33	1,948.33	3,950.24	1,082.87	1,439.10
		OR	Salem		10,497.18	3,194.89	1,236.45	687.24	1,271.20	2,049.73	518.48	806.59
			Salem	Store 13	10,497.18	3,194.89	1,236.45	687.24	1,271.20	2,049.73	518.48	806.59
		WA	Tacoma		8,728.50	2,213.11	706.54	751.35	755.22	1,658.05	457.88	504.46
			Tacoma	Store 17	8,728.50	2,213.11	706.54	751.35	755.22	1,658.05	457.88	504.46
		OR	Portland		6,780.35	1,704.66	484.44	543.09	677.13	1,900.51	564.39	632.51
			Portland	Store 11	6,780.35	1,704.66	484.44	543.09	677.13	1,900.51	564.39	632.51
		CA	Los Angeles		6,578.96	1,581.58	451.12	603.54	526.92	1,352.09	413.41	438.01
			Los Angeles	Store 7	6,578.96	1,581.58	451.12	603.54	526.92	1,352.09	413.41	438.01
		WA	Bremerton		6,497.72	1,521.96	493.76	580.57	447.63	1,807.54	541.17	360.83
			Bremerton	Store 3	6,497.72	1,521.96	493.76	580.57	447.63	1,807.54	541.17	360.83
		CA	San Diego		6,446.59	1,532.91	494.84	477.84	560.23	1,655.60	504.30	533.67
			San Diego	Store 24	6,446.59	1,532.91	494.84	477.84	560.23	1,655.60	504.30	533.67
		WA	Seattle		6,220.23	1,732.99	520.54	484.41	728.04	1,491.30	492.17	607.47
			Seattle	Store 15	6,220.23	1,732.99	520.54	484.41	728.04	1,491.30	492.17	607.47
			Spokane		5,750.03	1,485.00	366.27	576.29	542.44	1,311.16	424.28	389.60
			Spokane	Store 16	5,750.03	1,485.00	366.27	576.29	542.44	1,311.16	424.28	389.60
		CA	Beverly Hills		5,526.18	987.54	342.18	306.65	338.71	1,474.72	551.58	478.62
			Beverly Hills	Store 6	5,526.18	987.54	342.18	306.65	338.71	1,474.72	551.58	478.62

Figure 3-13 Sorting the Top West Coast Stores in Sales

3.2.4 Drilling Through to Details

The drill-through operation displays detailed transaction information for a given aggregated value.

To work with source data:

1. Click the number \$10,497.18 next to Store 13.



The drill-through table appears in a new page.

This is the default behavior. You can also display the drill-through table in the same page as the navigation table (see [Figure 3-34 on page 50](#)).

The drill-through table shows the underlying data in the database that is used to generate the summarized information in the navigation table. This is useful for validation of results. Drill-through data can also reveal interesting trends or anomalies. For example, you might identify a particular demographic that tends to make larger purchases.

As shown in the following figure, Chips are among the items making up the \$10,497.18 in Store Sales.

Foodmart Sample Analysis View

  Drill Through Table for [Store_Sales = 10,497.18]

Store_Name	Store_Sqft	Store_Type	Year	Quarter	Month	Week	Day	Product_Family	Product_Department	Product_Category	Product_Sub
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Chips
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Chips
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Chips
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Chips
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Cookies
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Cookies
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Cookies
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Cookies
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Cookies
Store 13	27694	Deluxe Supermarket	2006	Q1	1	3	7	Food	Snack Foods	Snack Foods	Cookies






Page 1/153  Goto Page  Rows/page 




Figure 3-14 Drill-through Table Showing High-value Items

Note that you can export the current set of source data to an Excel spreadsheet by clicking  at the top of the drill-through table. You can change the columns that are displayed and their sort order by clicking . Navigate through the paged data and control the number of rows per page using the controls at the bottom of the drill-through table.

3.2.5 Displaying Charts

Charts can provide more dramatic visual impact.

To view a chart of the example in [Figure 3-12 on page 35](#):

1. Close the drill-through table.
2. Click **Collapse All** .
3. Enable **Zoom on Drill** by clicking .
4. Zoom into USA.
5. Zoom into 2006.
6. Click **Show Chart** .

A default chart output appears.

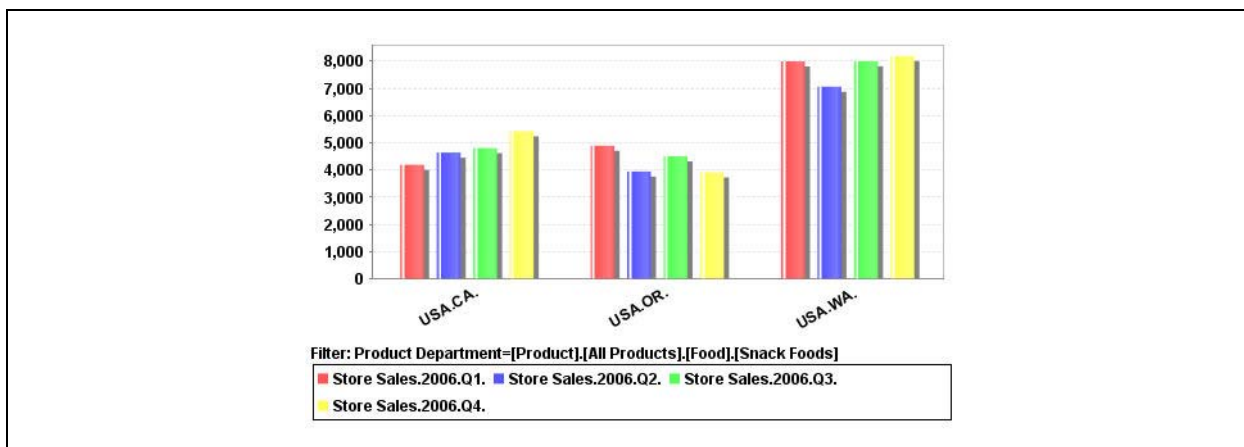


Figure 3-15 Default Chart for Product Sales by Time Across Store Locations

3.2.6 Exporting Output

OLAP views can be exported to Excel and PDF formats:

- Click **Output as Excel** to export the OLAP view to Excel.
- Click **Print as PDF** to export the OLAP view to a PDF.

The following figure shows typical Excel output:

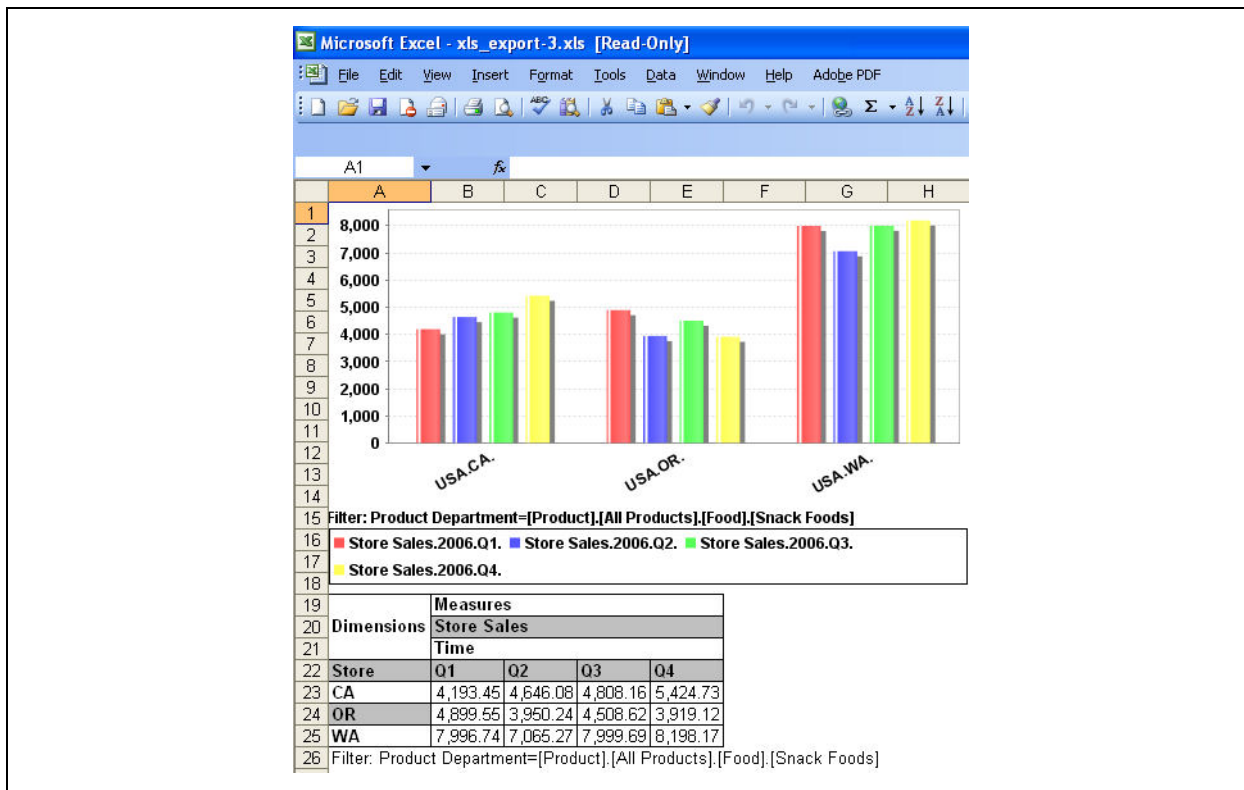


Figure 3-16 Typical Excel Output

3.3 Using the OLAP View Tools

Jaspersoft OLAP provides a graphical interface for analyzing multidimensional data. Its tool bar and navigation table provide all the controls necessary to analyze multidimensional data. This section explains your options.

3.3.1 Analysis Tool Bar and Navigation Table

The buttons are grouped by function: display, cube, chart, output, and save:

- The Display buttons configure dimension and navigation options.
- The Cube buttons mainly change cube contents.
- The Chart and Output buttons let you add a chart to the view and export the view to specific file formats.
- The Save buttons let you save your changes to the view.

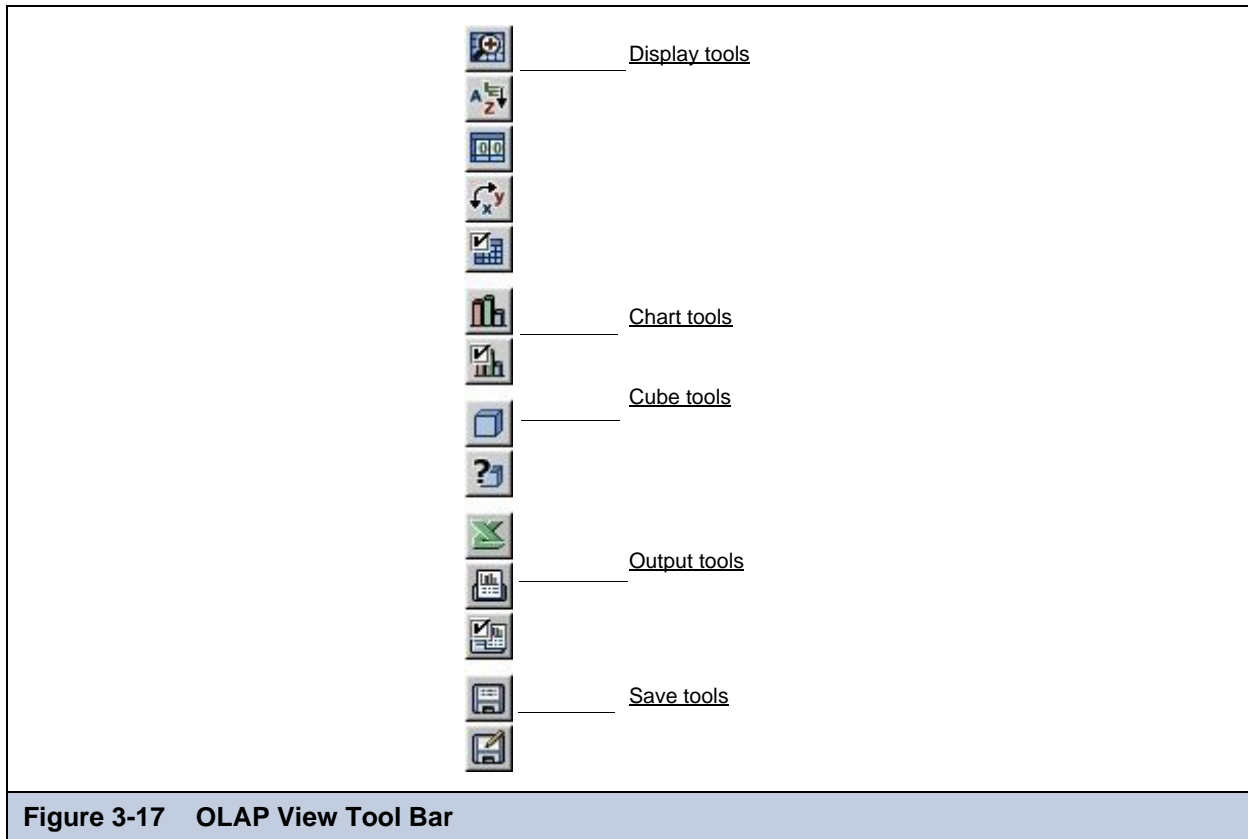


Figure 3-17 OLAP View Tool Bar

The navigation table displays rows and columns (that is, horizontal (X axis) and vertical (Y axis) elements) to organize information containing multiple logical dimensions (for example, Product, Customers, Time, and Store). The underlying data points, or facts, are summarized by the logical dimensions in terms of the measures, which are aggregated values and calculations, using standard functions such as sum, average, etc of the fact values, or more sophisticated calculated values such as year over year ratios and linear regressions.

Figure 3-18 shows a navigation table with three dimensions— Time, Store, and Promotion Media. Two of the dimensions (Promotion Media and Store) are placed on the vertical axis on the left; and the other dimension (Time) is placed on the horizontal axis along the top. The numbers at the intersection of each fiscal quarter and state indicate the store sales in that state during that time; this is the measure being analyzed.

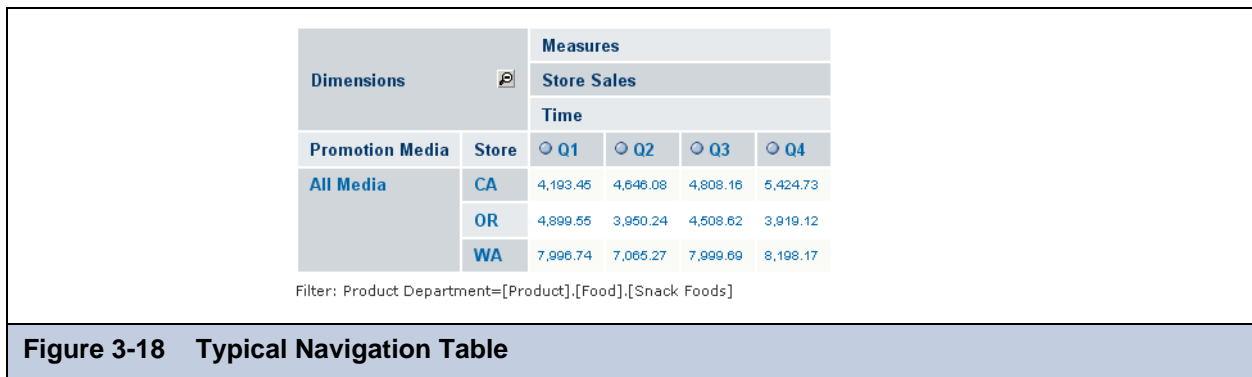



Figure 3-18 Typical Navigation Table

3.3.2 Cube Configuration

The data cube behind a navigation table can be configured by controls in the following dialogs in the Display Options group: Change Data Cube, Show MDX Query, and Sort Options. These options are described in the following sections.

3.3.2.1 Columns, Rows, and Filters

Click  to open the Change Data Cube dialog.

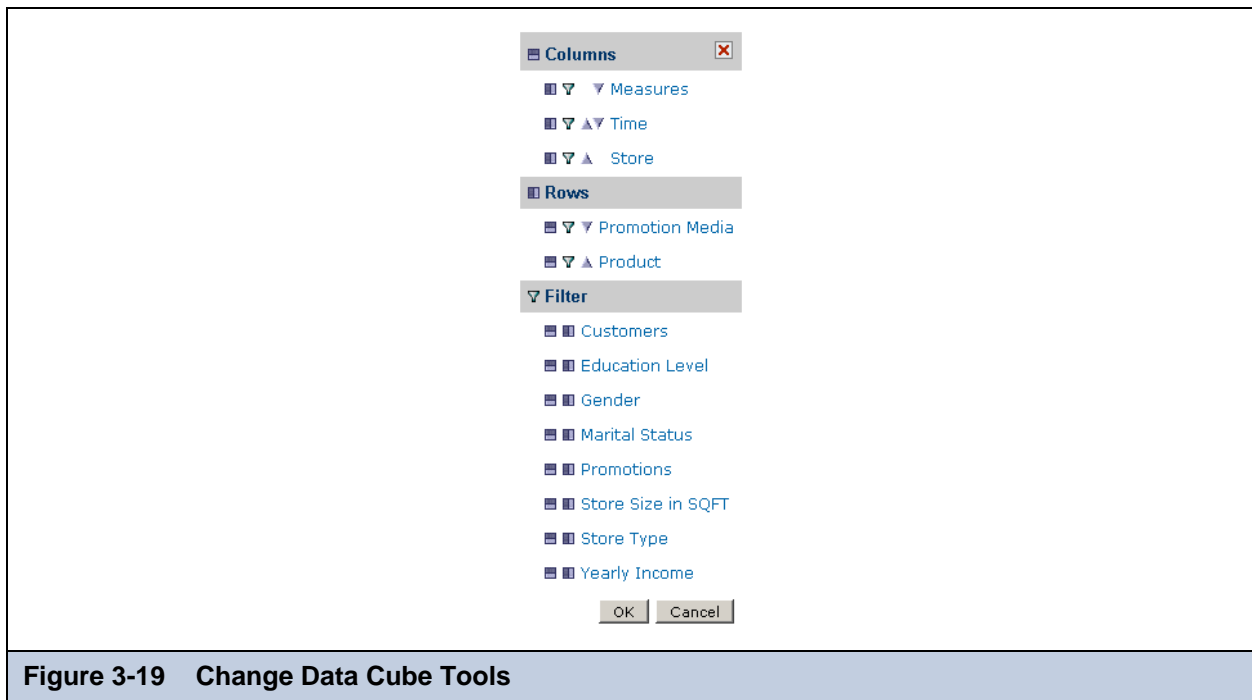


Figure 3-19 Change Data Cube Tools

The Change Data Cube dialog configures the information that appears in the navigation table. In the table, measures are organized as the columns on the right. The measures are aggregations (such as Store Sales) and each row is a summary of the measures in a different dimension, such as Store Sales by Store Country and Store State.

The navigation table can also be filtered by dimensions that are not displayed, for analyses like “Show Store Sales for June by Product and Store.” In this case, the filter would be Time = June, the rows would be Product and Store, and the columns would be Store Sales.

After making changes with the Change Data Cube tools, you must click **OK** for the selections to take effect.

Columns and Rows

The Columns section of the Change Data Cube dialog configures the measures of crosstab navigation tables. Measure columns are normally aggregations, such as *STORE SALES*, while measure rows are granularities of such aggregations, such as *STORE SALES* by *STORE COUNTRY*, *STORE STATE*, and *STORE CITY*.

The Dimensions section of the dialog configures the table's rows. Dimension rows are hierarchy levels representing the granularity of the hierarchies, such as Canada, Mexico, and USA for *STORE COUNTRY*, and CA, OR, and WA for *STORE STATE*. The row position of the dimensions can be changed by clicking the triangles, similar to how column positions can be changed.

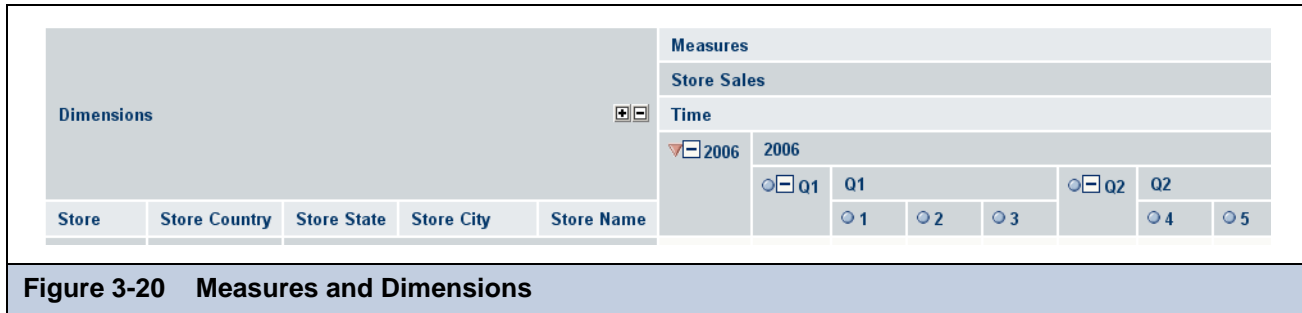


Figure 3-20 Measures and Dimensions

Measures can be selected, deselected, moved to a different column position, or moved to rows (pivoted).

To configure a measure:

1. Click the **Measures** hyperlink in the Change Data Cube dialog.

The Measures dialog appears.



Figure 3-21 Measures Dialog

2. Click a check box to select its measure.

To move a measure within its section of the dialog:

1. Click the blue dot icon next to the measure; the icon changes color to red. Other icons throughout the dialog change, as well, to signal that moving is enabled.

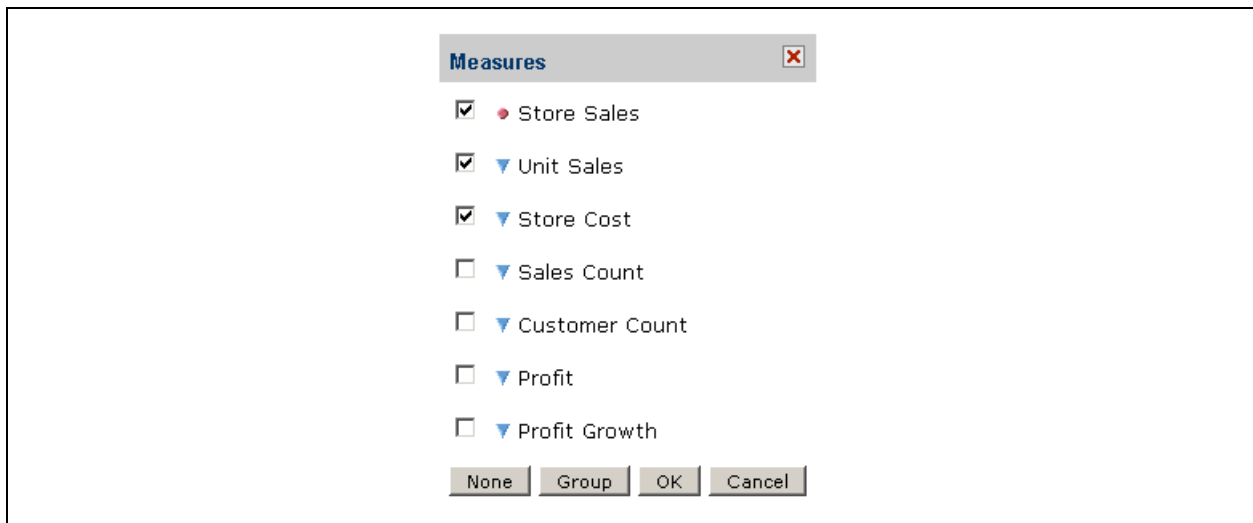


Figure 3-22 Measures Dialog with Moving Enabled

2. Click any triangle. The measure selected in **step 1**, Store Sales, moves down one position in the list.
3. Click a check box to indicate that the corresponding measure should be displayed in the view.
4. Click **Group** to collapse the list of measures into groups of 12 measures each.
The **Group** button changes to a **Flat** button. Click **Flat** to return to the expanded list.
5. Click **None** to clear the icons and your selections. This removes all measures from the view.

To move a dimension into the Columns or Rows section of the Change Data Cube dialog:




- Click  next to the dimension to move the data to a column.
- Click  next to the dimension to move it to a row.
- To configure a dimension, click the dimension in the dialog. For example, to configure the STORE dimension, click **STORE** in the Rows section, and the root level of the STORE dimension appears:



Figure 3-23 STORE Dimension

The root level of the STORE dimension contains a special member called ALL STORES. Clicking the check box next to the member selects all hierarchy levels of the STORE dimension. Clicking  displays the members in the next level of the STORE hierarchy. Select a lower-level member to limit the view to that data.

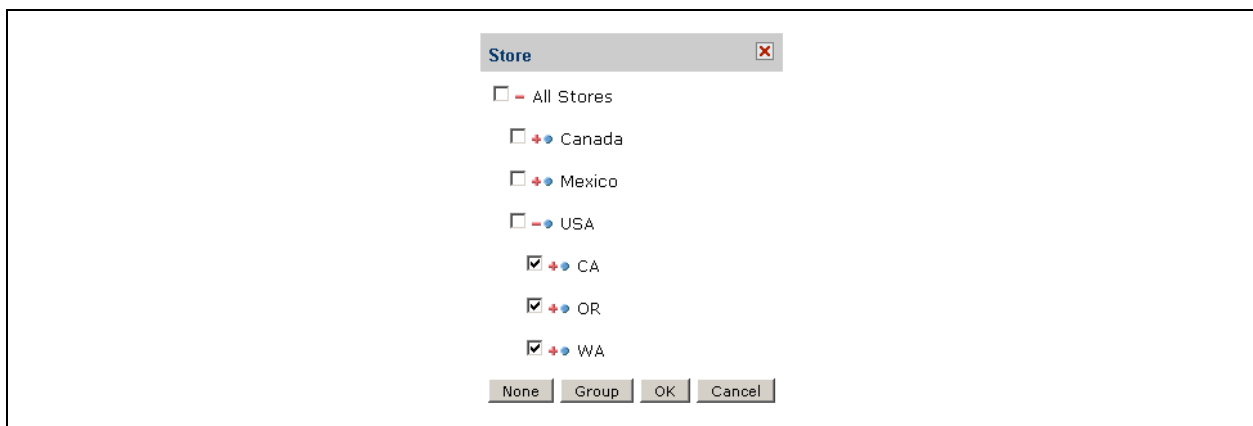



Figure 3-24 STORE Dimension

Filter

Dimensions can be used to filter the data in a navigation table.

To use a dimension as a filter:

1. In the Change Data Cube dialog, click  next to the dimension.
The dimension appears in the Filter section.

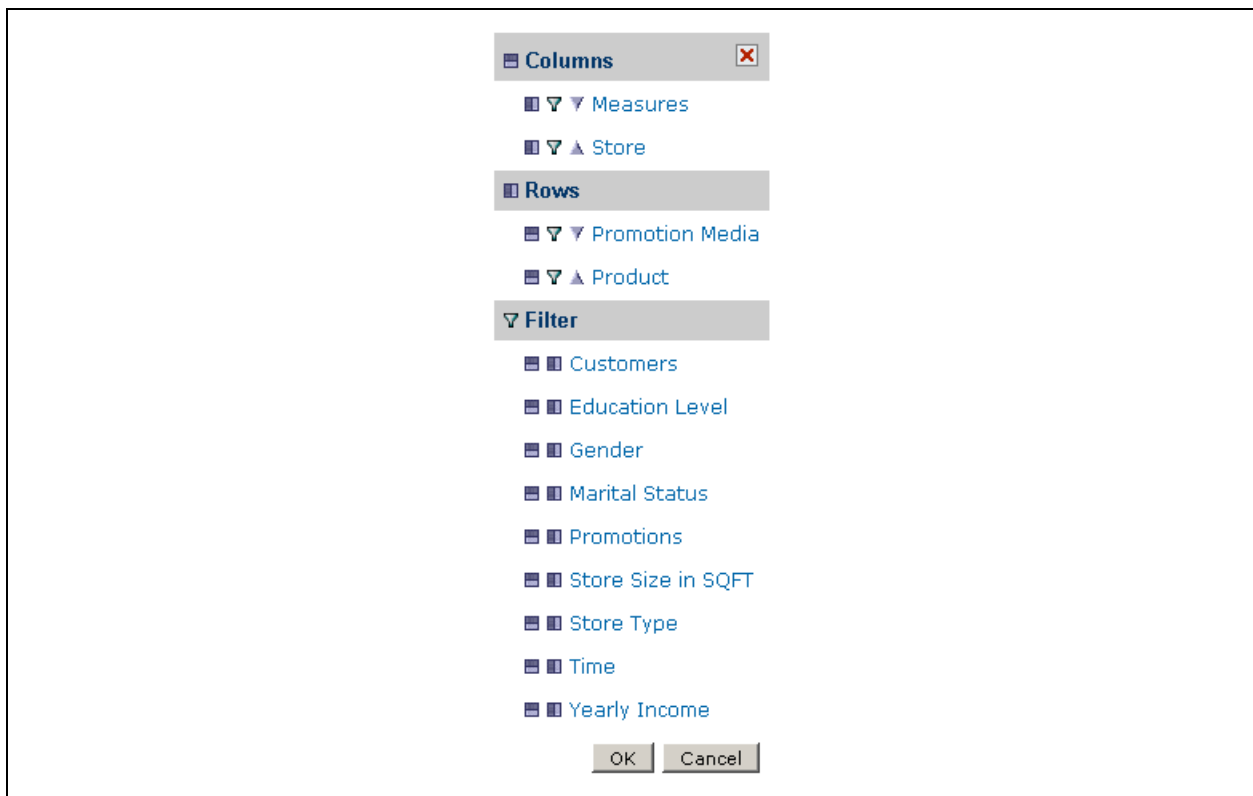


Figure 3-25 TIME Dimension as a Filter

2. Click the dimension.
The root level of the dimension appears. In this example, we're looking at the TIME dimension.

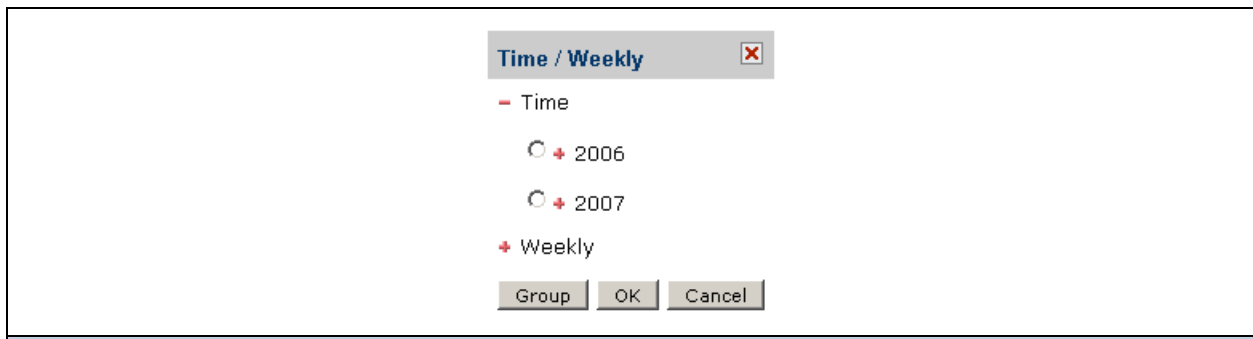


Figure 3-26 Selecting a Member from the TIME Dimension

3. Click **Group** to collapse the list of TIME measures into groups of 12 measures each. The **Group** button changes to a **Flat** button. Click **Flat** to expand the list of measures.



Only one member of a dimension can be used as a filter. Also, the member position in the dialog is fixed and cannot be changed.

3.3.2.2 Show MDX Query

The MDX Query Editor contains the MDX query that retrieves the contents of the navigation table. As you change the content of the navigation table, the MDX query is automatically updated. You can also change the contents of the navigation table by

changing the MDX in the editor. Click  to open the MDX Query Editor.



Figure 3-27 MDX Query Editor

An MDX query consists of data sets, query scope, and filter specifications:

- A SELECT statement determines the data sets that will populate the columns (x-axis) and rows (y-axis) of the navigation table. The SELECT statement includes the measures to use as columns and rows. The query in this example specifies data sets in terms of:
 - [Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales] as columns; [Promotion Media].[All Media] and [Product].[All Products] as rows.
 - The FROM clause specifies the cube that is queried. You can query only one cube at a time.
 - The WHERE clause uses dimensions to constrain the data sets retrieved by the query, that is, the clause specifies the filters that screen the data the query returns. In the example, [TIME].[2006] is the filter.

Click **Apply** to update the navigation table in the OLAP view. The system validates the query and updates the navigation table. Click **Revert** to discard all changes.

For a reference to the MDX query language, see <http://msdn2.microsoft.com/en-us/library/ms145506.aspx>.



The rest of the examples in this chapter start with the query in **Figure 3-27** above. To re-create the examples yourself, start with that query.

3.3.2.3 Sort Options

The sort options include the following:

- Sort across cube hierarchy.
- Start sorting in descending order.
- Display only the first N rows.

To use the sort options, first click  to open the Display Options dialog:

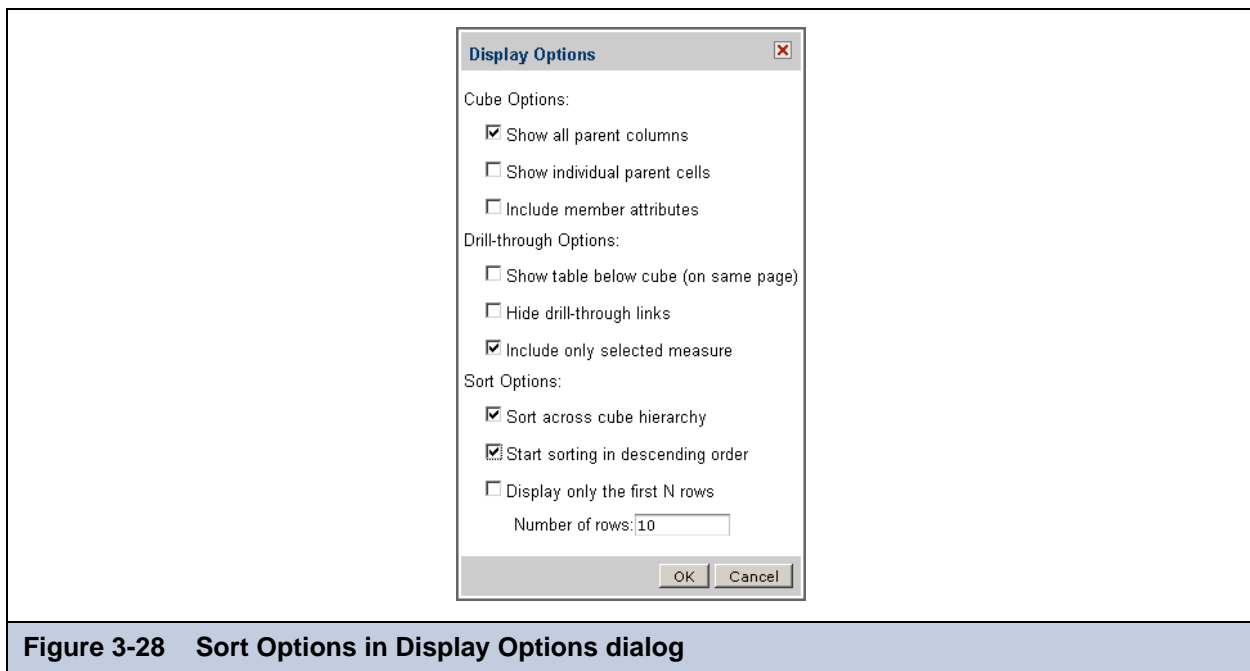


Figure 3-28 Sort Options in Display Options dialog

Then make your selections in the Sort Options section of the dialog and click **OK**.

The following sections explain the options.

Sort across cube hierarchy


Generally, dimension members are sorted within their parent member. Sort across cube hierarchy changes the navigation table such that member measures are sorted without regard to the hierarchies that contain them.

The STORE STATE categories CA, OR, and WA are examples of hierarchy boundaries; STORE CITY lists the hierarchy members. Sorting STORE STATE across hierarchy boundaries sorts the STORE CITY members regardless of the hierarchy boundaries. Put another way, cities are sorted without regard to their states.

Dimensions				Measures		
Store	Store Country	Store State	Store City	Store Sales	Unit Sales	
All Stores				565,238.13	266,773	
All Stores	USA			565,238.13	266,773	
	USA	CA			159,167.84	74,748
		CA	+ Alameda			
			+ Beverly Hills	45,750.24	21,333	
			+ Los Angeles	54,546.28	25,663	
			+ San Diego	54,431.14	25,635	
			+ San Francisco	4,441.18	2,117	
		OR			142,277.07	67,659
		OR	+ Portland	55,058.79	26,079	
			+ Salem	87,218.28	41,580	
		WA			263,793.22	124,366
		WA	+ Bellingham	4,739.23	2,237	
			+ Bremerton	52,896.30	24,576	
			+ Seattle	52,644.07	25,011	
			+ Spokane	49,634.46	23,591	
			+ Tacoma	74,843.96	35,257	
			+ Walla Walla	4,705.97	2,203	
			+ Yakima	24,329.23	11,491	

Filter: Year=[Time].[2006]

Figure 3-29 Sorting Across the Cube Hierarchy

Click  to toggle between sort across and sort within hierarchy boundaries.

Start sorting in descending order

By default, data are sorted in ascending order. To sort in descending order, select the **Start sorting in descending order** check box.

Display only the first N rows

Display only the first N rows limits the display to either the top-most or bottom-most *n* rows, depending on the sort order. The user can specify an integer for *n*, which defaults to 10. Limiting the display to a certain number of rows can help with performance or narrow the view to highlight records at the extremes of your dataset; for example, it can be used to highlight the top 10 sellers or the bottom 10 prices. Clear the check box to disable the row limitation.

3.3.3 Dimension Column Layout

The layout of the dimension columns in a navigation table is configured using these options:

- Edit Display options:
 - Show all parent columns
 - Show individual parent cells
 - Include member attributes
- Hide Empty Rows/Columns

- Swap Axes

3.3.3.1 Show all parent columns

The **Show all parent columns** check box displays a header for every dimension column. When the check box is cleared, all the columns are collapsed into one; the rows of that column have the same hierarchy as the separate columns. For example, in **Figure 3-30**, the All parent view has separate columns for STORE, STORE COUNTRY, and STORE STATE. The No parent view has only one column; the topmost header is STORE, then ALL STORES, then USA, then the individual states.

Show all parent columns is often used with **Zoom on Drill** to provide complete context after zooming in.

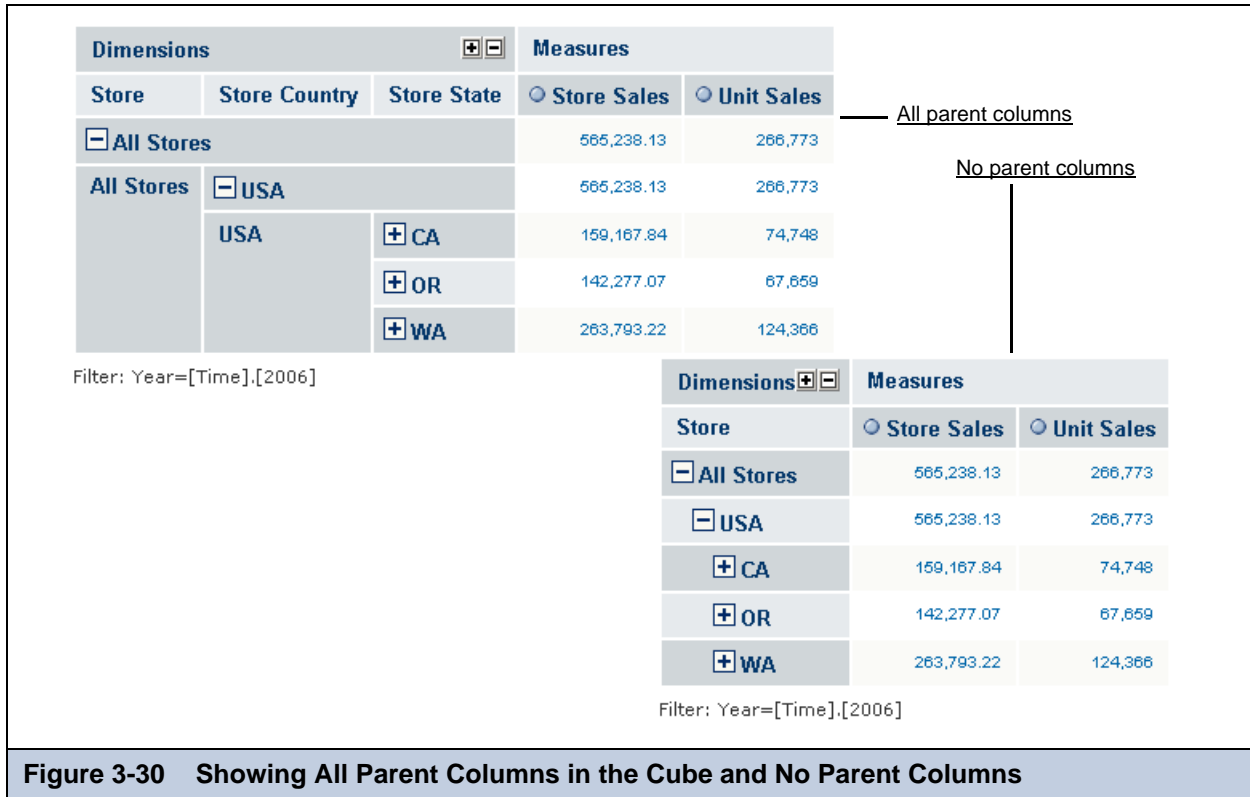


Figure 3-30 Showing All Parent Columns in the Cube and No Parent Columns

3.3.3.2 Show individual parent cells

The **Show individual parent cells** check box determines whether redundant dimension hierarchy member labels are displayed or hidden. The check box takes effect when the **Show all parent columns** check box is checked. The following figure shows the example in **Figure 3-30** with **Show individual parent cells** selected. This view is useful when outputting the table to an Excel spreadsheet so that headers appear on every page. It also prevents cell-spanning and the empty cells that result.

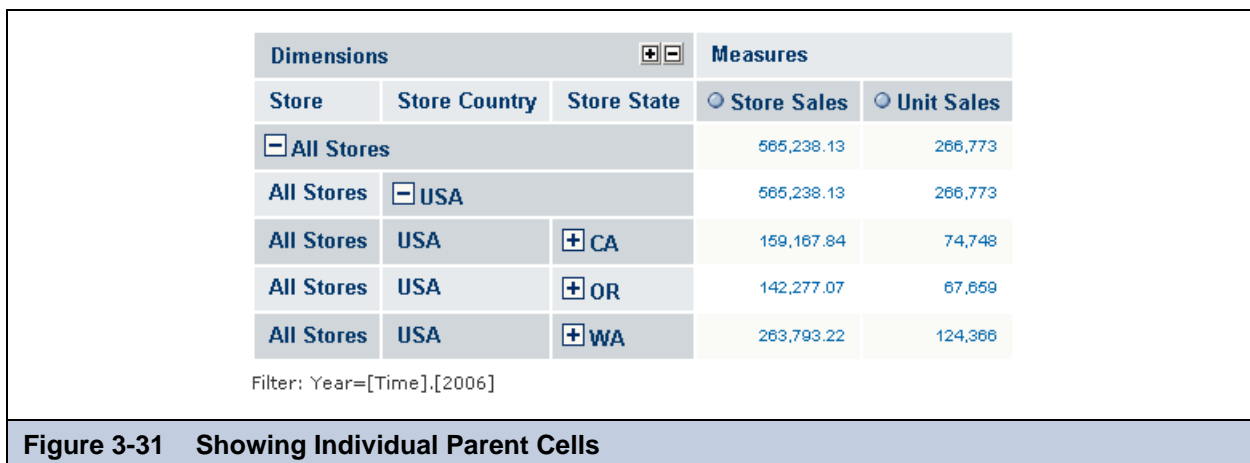





Figure 3-31 Showing Individual Parent Cells

3.3.3.3 Hide Empty Rows/Columns

Clicking  toggles between showing and hiding rows that do not have a value for the selected measures. By default, all rows of measures display, even when they have no value.

3.3.3.4 Swap Axes

Clicking  changes the orientation of a navigation table by switching the columns with the rows—each column becomes a row, and each row becomes a column.

Suppose a table contains five dimension rows and two measure columns. Clicking  changes the orientation of the table so it has one measure row and five dimension columns.

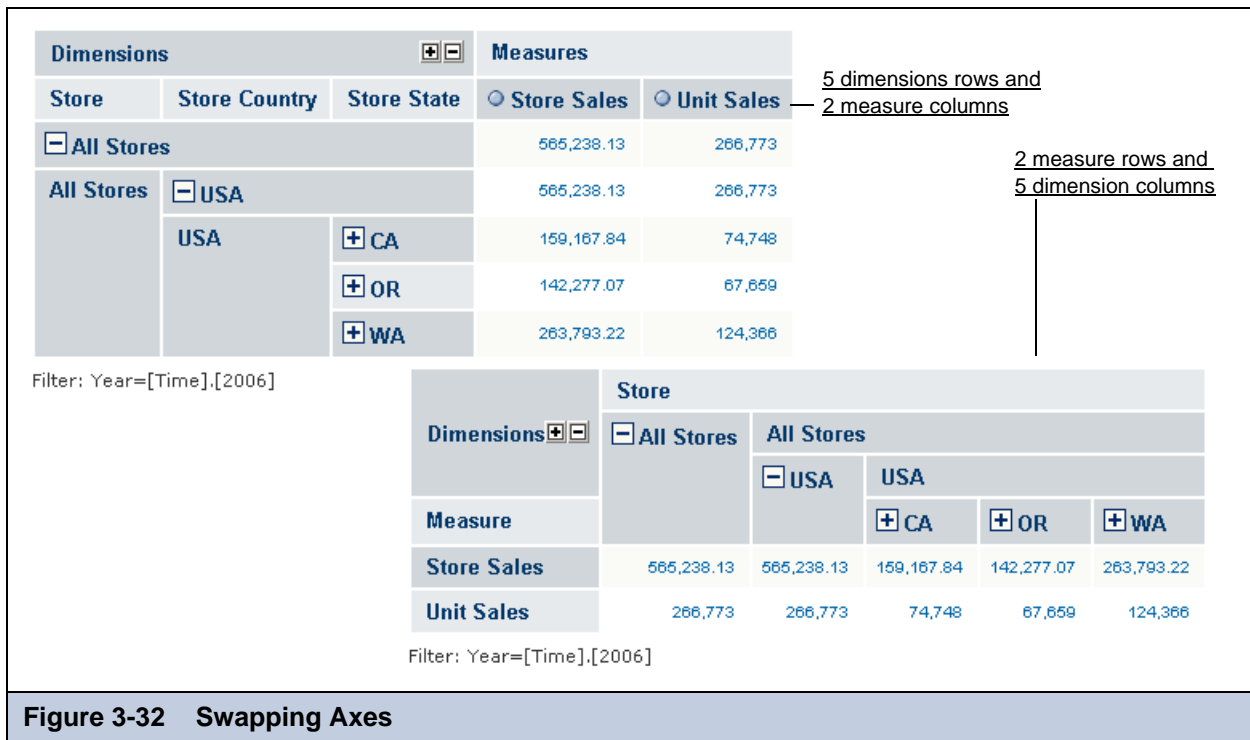


Figure 3-32 Swapping Axes

3.3.4 Navigation Table Controls

The navigation table displays the data from the OLAP view as rows and columns. The table controls adjust the display to help you understand the data. Three controls apply to dimension members:


- **Expand Member**
- **Expand Position**
- **Zoom on Drill**


Drill-through applies to measures.

3.3.4.1 Expand Position

Expand Position displays the child members of the selected member.

To expand a position, click  in its header; to collapse it, . **Expand Position** affects only the member that you click; the sibling members remain as they were.

You can also expand the columns one at a time by clicking  in the icon group at the top of the navigation table. The columns expand sequentially from left to right. In the example, STORE would be expanded first, then STORE COUNTRY, STORE STATE, STORE CITY, and finally STORE NAME.

Clicking  in the icon group collapses *all* of the columns, not one column at a time.

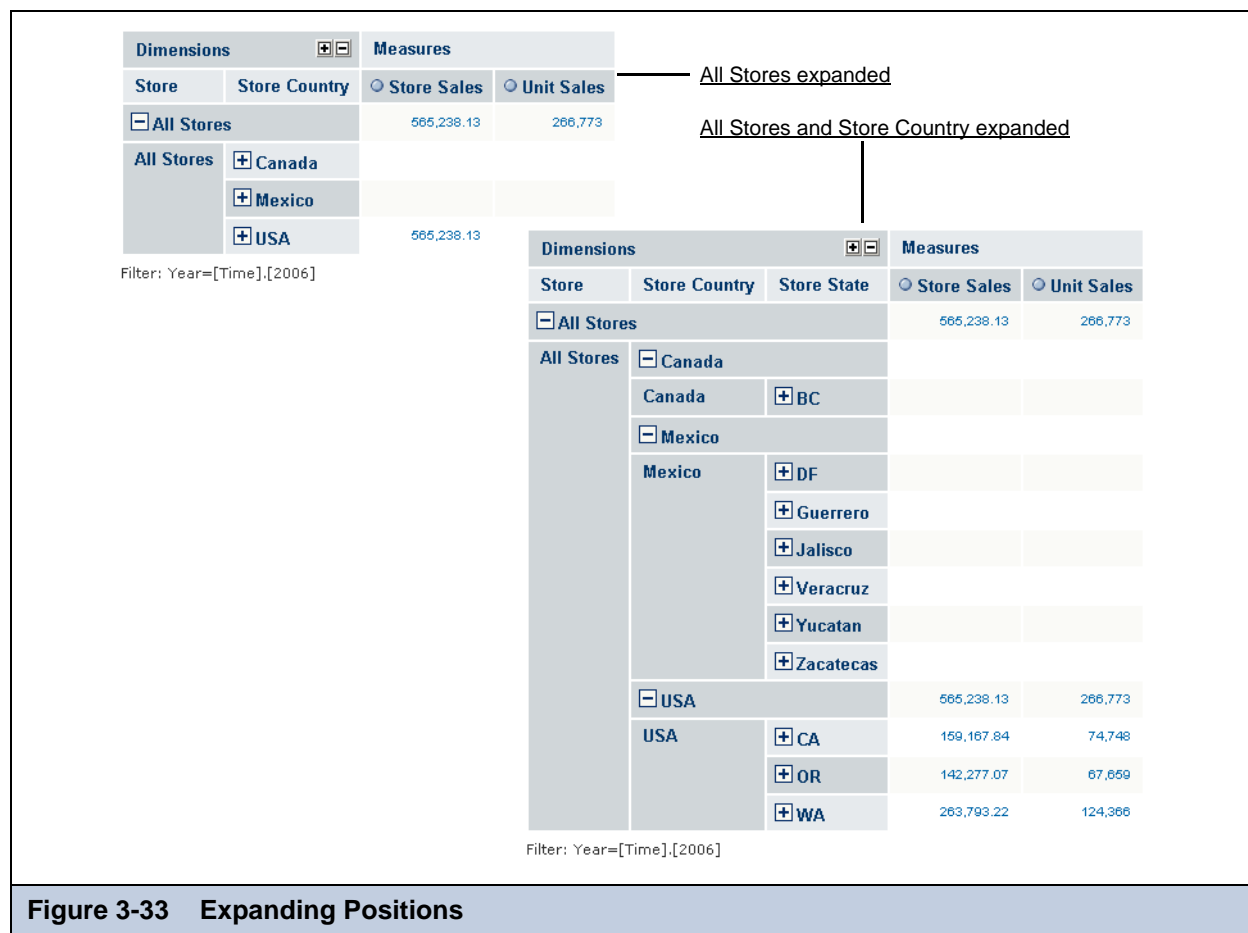


Figure 3-33 Expanding Positions

3.3.4.2 Zoom on Drill

Zoom on Drill displays the dimension members that compose the current member. For instance, zooming on Products displays the members in Product— Drink, Food, and Non-Consumable. Likewise, zooming on Drink displays the members in Drink— Alcoholic Beverages, Beverages, and Dairy.

To zoom on a member, select . The **Zoom on Drill** cursor  appears. Click the cursor on the member.

By default, when you select **Zoom on Drill**, the **Show all parent columns** cube option is turned on, as well; this keeps the member hierarchy in the view.

3.3.4.3 Drill-through

Drill-through displays the data underlying a Measures value, that is, the data that produced the Measures value, provided the value is aggregated from other values. For instance, the total sales figures for a store are aggregated from the sales figures of everything sold in the store. The data for one sale of one item is not aggregated. Measures values are usually aggregated. By default, the data contains every column in the source data.

Click any aggregated value in the Measures columns to display the drill-through table for that value. The following figure shows part of the drill-through table for the measure 74,748, which is the unit sales measure for CA (California). Notice that the table is very large; it has 2,445 pages. At ten rows to a page, that's 24,450 rows.

To disable drill-through, open the Display Options dialog (**Figure 3-28 on page 45**) and select **Hide Drill-through links**.


Drill Through Table for [Unit_Sales = 74,748]											
Store_State	Store_City	Store_Name	Store_Sqft	Store_Type	Year	Quarter	Month	Week	Day	Product_Family	Product_Department
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Drink	Dairy
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2006	Q1	1	2	1	Food	Baked Goods

Page 1/2,445 ▶▶ Goto Page Rows/page

Figure 3-34 Drill-through Table for Unit Sales

In the drill-through tables, these controls are available:

- **Sorting.** Rows in the drill-through table can be sorted based on a given column. To sort by a column, use the sort icons next to that column’s header: ▲▼. Clicking that column’s ▼ resets the sort mode to natural order, that is, the record order in the database, which is indicated by ●.
- **Forward/Backward.** Click the left and right arrows to go to another page.
- **First/Last.** Click the double arrows to go to the first or last page.
- **Go To Page.** Enter a number and click the icon to go to the specified page.
- **Rows per Page.** Enter a number and click the icon to set the number of rows per page. Same as **Page size** above.

Notice that there are 2,445 pages in the table. To filter the table and reduce its size, use the Edit Properties dialog. To open the dialog, click  at the top-left corner of the drill-through table.

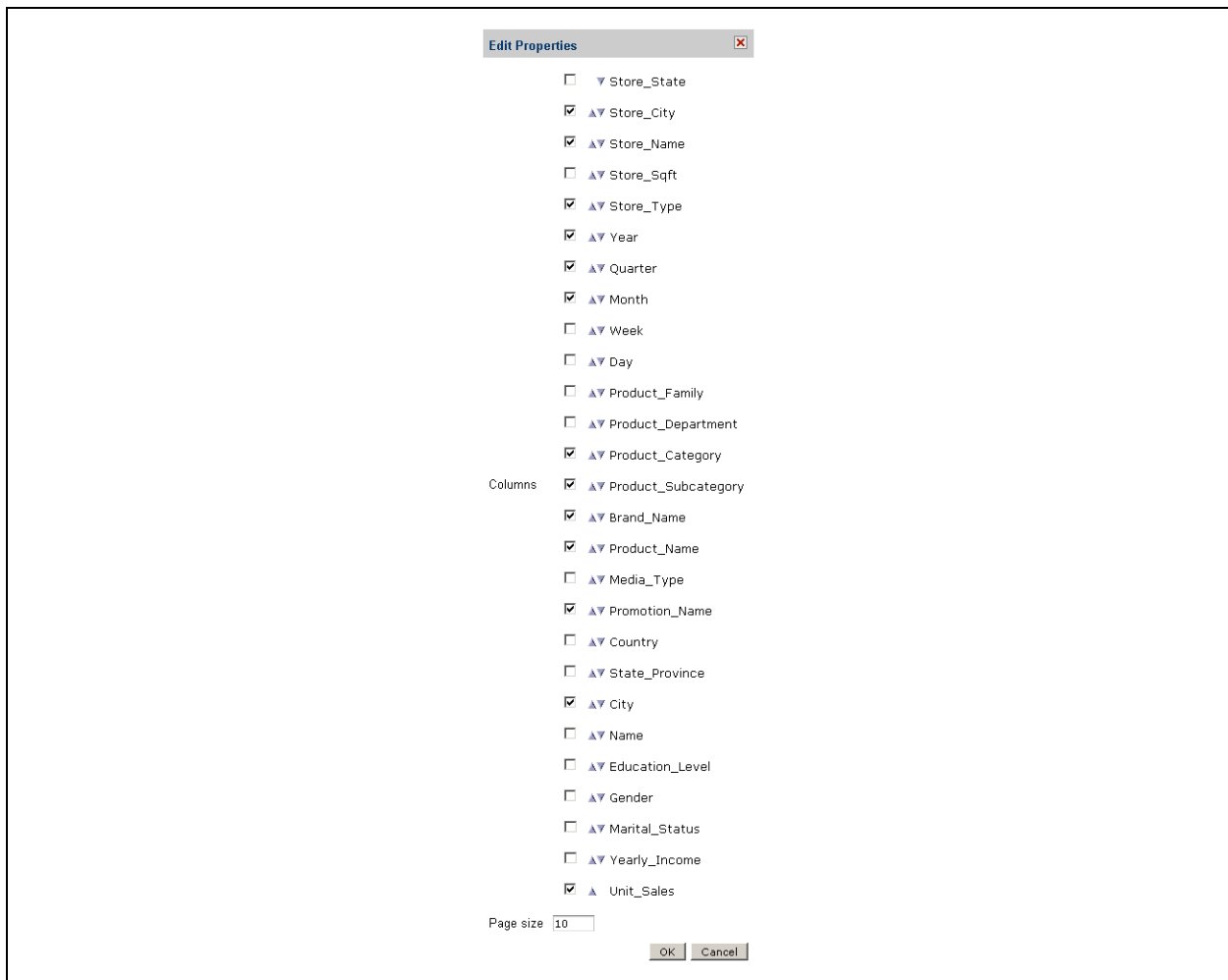



Figure 3-35 Edit Properties Dialog for a Drill-through Table

The Edit Properties dialog has two controls:

- **Columns.** To include a column in the table, select its check box. Use the up-down triangles ▲▼ to arrange the column in the table. Clicking ▲ moves the column to the left in the table; clicking ▼ moves the column to the right. To hide a column from view, clear its check box.
- **Page size.** Specifies the number of rows on a page. Active only when **Enable paging** is selected.

3.3.5 Charts

Charts are a graphical representation of the navigation table.

To select the chart type and to configure the chart, open the Chart Options dialog by clicking **Edit Chart Options**  (Figure 3-36).

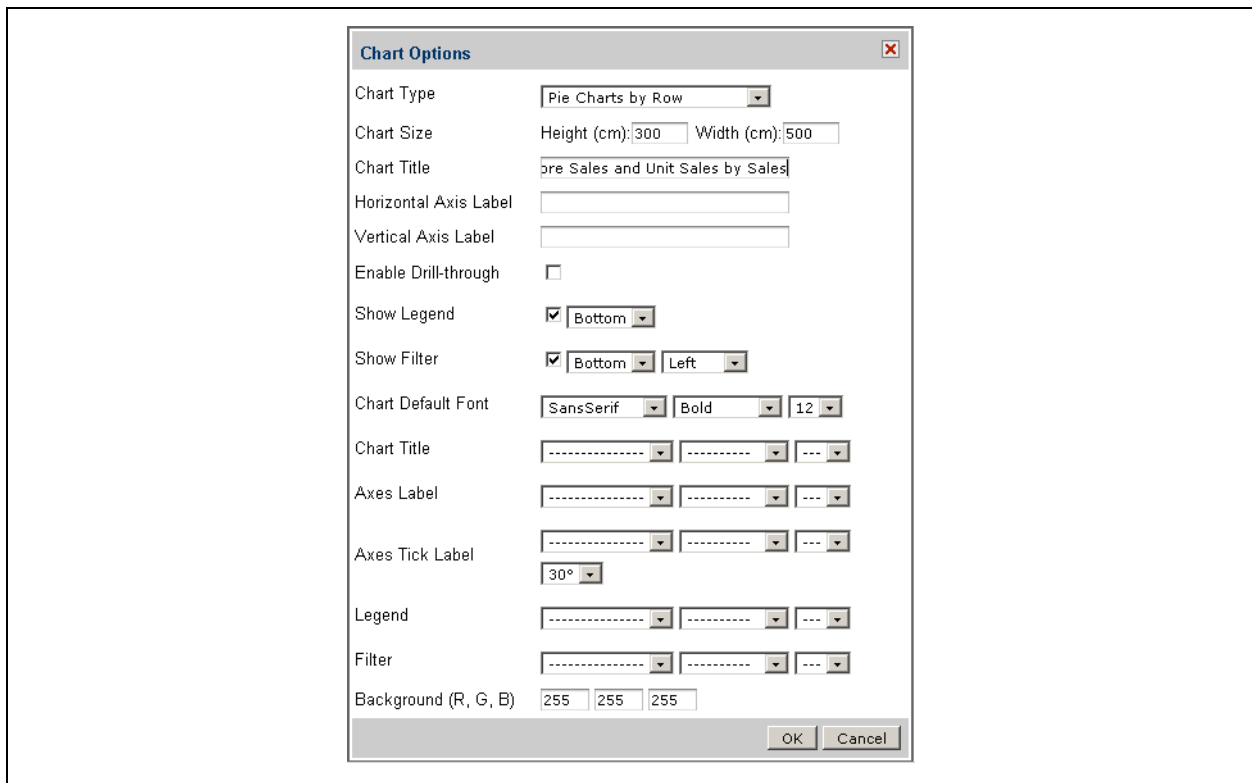


Figure 3-36 Chart Options Dialog


Then, to display the chart, click **Show Chart** .

Figure 3-37 shows the chart selected in the Chart Options dialog. It is a pie chart of the Store Sales and Unit Sales measures for three states; the size of the pie slices indicates the ratio of one state's sales compared to the total sales for all states.

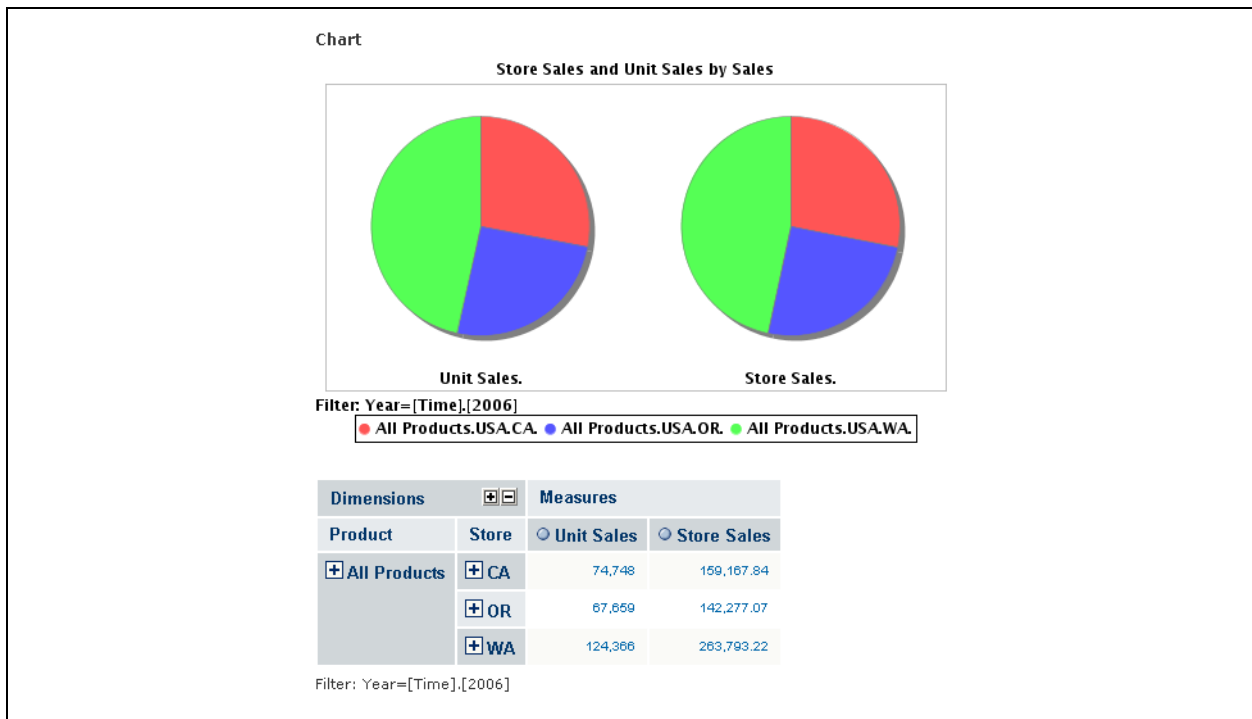




Figure 3-37 Pie Chart Showing Store Sales and Unit Sales by State

The Chart Options dialog has these options:

- **Chart Type.** Select a chart type from the drop-down of available chart types. The default type is Vertical Bar.
- **Chart Size.** Set the chart height and width in terms of pixels. The default chart height and width are 300 by 500 pixels.
- **Chart Title.** Enter the title of the chart.
- **Horizontal Axis Label.** Enter the label to display on the horizontal axis of the chart.
- **Vertical Axis Label.** Enter the label to display on the vertical axis of the chart.
- **Enable Drill-through.** Select the check box to enable drill-through in the chart. By default, drill-through is disabled.
- **Show Legend.** Select the check box to show the chart legend, and select its position from the drop-down. By default, the legend is displayed and its position is Bottom. Clear the check box to hide the legend.
- **Show Filter.** Select the check box to show the slicer, and select the position and side for it from the drop-downs. By default, the filter is displayed, its position is Bottom, and its side is Left. Clear the check box to hide the filter.
- **Chart Default Font.** Select the chart's default font name, style, and size. The default is SansSerif Bold 12.
- **Chart Title.** Select the font name, style, and size of the chart title.
- **Axes Label.** Select the font name, style, and size of the chart title of the chart axes, or use the default chart font.
- **Axes Tick Label.** Except for pie charts, you can select an Axes Tick Label font, or use the default chart font. The default text angle is 30°.
- **Legend.** Select the font name, style, and size of the chart legend if font information isn't specified, Jaspersoft OLAP uses the default chart font.
- **Filter.** Select the font name, style, and size of the filter label; if font information isn't specified, Jaspersoft OLAP uses the default chart font.
- **Background Color.** Set the chart background color in terms of Red, Green, and Blue values. The default value for all three colors is 255, which creates white.

3.3.6 Output Formats

Jaspersoft OLAP provides two output formats for the navigation table—an Excel spreadsheet and a PDF file:

- To select Excel output, click .
- To select PDF output, click .

You can view the output before saving it. Output properties can be configured in the Edit Output Options dialog. To open the

dialog, click **Edit Output Options** .

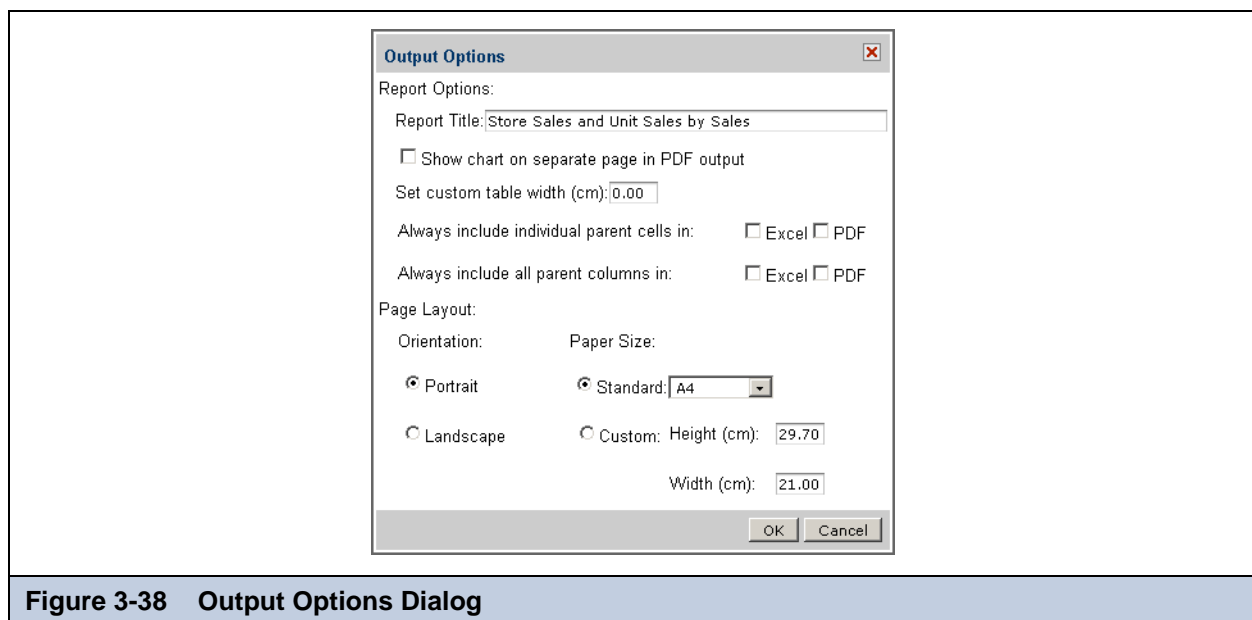


Figure 3-38 Output Options Dialog

These report options are provided:

- **Report Title.** Enter text to display as the title on above the navigation table.
- **Chart on separate page.** For PDF output only, select the check box to display the chart on a separate page. By default, the check box is unchecked. This setting is ignored if output is Excel.
- **Table Width.** Select the check box and enter a decimal number to set the table width in centimeters (cm). If the check box is unchecked, the table width is automatically set. By default, the check box is empty.
- **Include parent cells.** Select the check box to break spans and display redundant dimension hierarchy member labels in the output.
- **Include parent columns.** Display all dimension column headers in the output.

These are the page layout options:

- **Orientation.** Select a page orientation from the drop-down. By default, the orientation is Portrait.
- **Page size.** Select a page size from the drop-down. By default, the page size is A4.
- **Custom size.** Enter a report height and width in centimeters. By default, the width is 29.70 cm and the height is 21.00 cm. Setting both the height and width to 0 is equivalent to setting the page size to A4.

3.3.7 Save Options

After changing an OLAP view to suit your needs, save it by clicking one of the save buttons:

- **Save View.** Saves the view using its current name (as entered in 3.1, “OLAP Views,” on page 27). This overwrites the file of that name.
- **Save View As.** Saves the view with a different name.



Jaspersoft OLAP only saves the drill-through table if it is displayed on the same page as the navigation table. For more information, refer to 3.2.4, “Drilling Through to Details,” on page 36.

3.4 Troubleshooting Jaspersoft OLAP

Troubleshooting can be organized around the major activities in Jaspersoft OLAP—creating resources, creating and modifying OLAP views, loading OLAP views, expanding hierarchy positions and members, and drilling through. Most error messages are typically associated with a single cause and have a single corresponding resolution. However, some messages have multiple causes and may occur more frequently.

The following sections list the error messages and explain their causes.

3.4.1 Common Error Messages

Messages with Multiple Possible Causes	
Message	Possible Causes
<pre>com.jaspersoft.jasperserver.api.JSException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: Error while creating SQL dialect</pre>	Invalid database name specified in the data source URL. Invalid port number specified in the data source URL. Invalid hostname specified in the data source URL. Invalid database driver name. Invalid username and/or password.
<pre>com.jaspersoft.jasperserver.api.JSException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: while parsing catalog null</pre>	Aggregator of Measure in Cube definition is invalid. Data type of Level in Hierarchy definition is invalid. Level type of Level in Hierarchy definition is invalid

Messages with Multiple Possible Causes, continued	
Message	Possible Causes
com.jaspersoft.jasperserver.api.JSException: com.tonbeller.jpivot.olap.model.OlapException: Failed to parse query 'xxxxxx'	<p>ForeignKey of DimensionUsage in Cube definition is invalid.</p> <p>Format in measure definition is invalid.</p> <p>Column of Measure in Cube definition is invalid.</p> <p>Column of Dimension definition is invalid.</p> <p>The MDX query xxxxxx of the OLAP view does not match with the Mondrian schema with cube yyyy of the OLAP connection.</p>
An error occurred while getting property 'result' from an instance of class com.tonbeller.jpivot.tags.OlapModelProxy	<p>Dimension table name is invalid.</p> <p>Column of Level in dimension hierarchy is invalid.</p> <p>parentColumn of Level in dimension hierarchy is invalid.</p> <p>nameColumn of Level in dimension hierarchy is invalid.</p> <p>parentColumn of Closure in dimension hierarchy level is invalid.</p> <p>childColumn of Closure in dimension hierarchy level is invalid.</p> <p>Table of Closure in dimension hierarchy level is invalid.</p> <p>Mondrian schema does not match data source for the Mondrian connection.</p>
The following is not a valid MDX Query: Failed to parse query 'xxxxxx'	<p>Format in measure definition is invalid.</p> <p>The MDX query xxxx of the OLAP view does not match with the Mondrian schema with cube yyyy of the OLAP connection.</p> <p>ForeignKey of DimensionUsage in Cube definition is invalid.</p> <p>Column of Measure in Cube definition is invalid.</p> <p>Column of Dimension definition is invalid.</p>
The following is not a valid MDX Query: Internal error: Error while creating SQL dialect	<p>Invalid database name specified in the data source URL.</p> <p>Invalid port number specified in the data source URL.</p> <p>Invalid hostname specified in the data source URL.</p> <p>Invalid database driver name.</p> <p>Invalid username and/or password.</p>
The following is not a valid MDX Query: Internal error: while parsing catalog null	<p>Aggregator xxxx of Measure in Cube definition is invalid.</p> <p>Data type xxxx of Level in Hierarchy definition is invalid.</p> <p>Level type xxxx of Level in Hierarchy definition is invalid.</p>
com.jaspersoft.jasperserver.api.JSException: No OLAP Model name	The system is timed out after a long pause (usually about 30 min.) while interacting with an OLAP view. In this case, log out and log in again.

3.4.2 Messages that Occur When Creating and Modifying OLAP Views

Messages about Creating and Modifying OLAP Views			
Resource	Error Message	Possible Causes	Resolution
Data source	The following is not a valid MDX Query: Internal error: Error while creating SQL dialect	Invalid database name specified in the data source URL.	Check the database name and make sure a valid database name is used in the data source URL.
		Invalid port number specified in the data source URL.	Check the database port number and make sure a valid port number is used in the data source URL.
		Invalid hostname specified in the data source URL.	Use a valid hostname.
		Invalid database driver name.	Use a valid database driver name.
		Invalid username and/or password.	Use a valid database username and password. Then re-start JasperReports Server.
	The following is not a valid MDX Query: Communications link failure due to underlying exception: ** BEGIN NESTED EXCEPTION ** java.net.SocketException MESSAGE: java.net.ConnectException: Connection refused	No database connection.	Check database connection and re-start database server if necessary. Then re-start JasperReports Server.
	The following is not a valid MDX Query: Failed to parse query 'xxxxxxx'	ForeignKey of DimensionUsage in Cube definition is invalid.	Use a valid DimensionUsage foreignKey. Then reload the schema.
		Format in measure definition is invalid.	Use a valid format for the measure. Then re-start JasperReports Server and reload the schema.
		Column of Measure in Cube definition is invalid.	Use a valid database column for the measure. Then re-start JasperReports Server and reload the schema.
		Column of Dimension definition is invalid.	Use a valid database column for the dimension. Then re-start JasperReports Server, and reload the schema.

Messages about Creating and Modifying OLAP Views, continued			
Resource	Error Message	Possible Causes	Resolution
Data source, continued	The following is not a valid MDX Query: Internal error: while parsing catalog null	Aggregator xxxx of Measure in Cube definition is invalid.	Use one of { sum, count, min, max, avg, distinct count } as aggregator in the Measure. Then re-start JasperReports Server and reload the schema.
		Data type xxxx of Level in Hierarchy definition is invalid.	Use one of { String, Numeric } as data type in the Level. Then stop and re-start JasperReports Server and reload the schema.
		Level type xxxx of Level in Hierarchy definition is invalid.	Use one of { Regular, TimeYears, TimeQuarters, TimeMonths, TimeWeeks, TimeDays } as level type in the Level. Then re-start JasperReports Server and reload the schema.
OLAP schema	The following is not a valid MDX Query: Mondrian Error:Internal error: Cannot find shared dimension 'xxxx'	Source of DimensionUsage xxxx is invalid.	Use a valid dimension name for DimensionUsage source. Then reload the schema.
	The following is not a valid MDX Query: Failed to parse query 'yyyyyy'	Definition yyyyyy of dimension hierarchy dddd in cube cccc is incorrect.	Correct the hierarchy definition. Then re-start JasperReports Server and reload the schema.
OLAP Client connection	The following is not a valid MDX Query: Failed to parse query 'xxxxxx'	The MDX query xxxxxx of the OLAP view does not match with the Mondrian schema with cube yyyy of the OLAP connection.	Make sure the MDX query matches with the Mondrian schema of the OLAP connection for the OLAP view.

3.4.3 Error Messages When Creating Other Objects

The following error messages may occur during the creation of data sources, Mondrian schemas, and OLAP connections:

Messages about Creating Objects			
Resource	Error Message	Possible Causes	Resolution
Data source	org.springframework.webflow.ActionExecutionException : ... Resource "xxxxxx" already exists.	A data source with the name xxxxx already exists in the repository.	Choose a unique data source name.
OLAP schema	xxxx The above file name is invalid.	Schema file xxxx does not exist.	Choose a valid file name.

3.4.4 Messages that Occur When Loading OLAP Views

Messages about Loading OLAP Views			
Resource	Error Message	Possible Causes	Resolution
Data source	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: Error while creating SQL dialect	Invalid database name specified in the data source URL.	Check the database name and make sure a valid database name is used in the data source URL.
	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: Error while creating SQL dialect	Invalid port number specified in the data source URL.	Check the database port number and make sure a valid port number is used in the data source URL.
	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: Error while creating SQL dialect\	Invalid hostname specified in the data source URL.	Use a valid hostname. Then valid the OLAP view.
	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: Error while creating SQL dialect	Invalid database driver name.	Use a valid database driver name.
	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: Error while creating SQL dialect	Invalid username and/or password.	Use a valid database username and password. Then stop and re-start JasperReports Server, and reload the schema.
	org.springframework.transaction.CannotCreateTransactionException: Could not open Hibernate Session for transaction; nested exception is org.hibernate.exception.JDBCConnectionException: Cannot open connection	No database connection.	Check database connection, and re-start database server if necessary. Then stop and re-start JasperReports Server, and reload the schema.
	OLAP schema	com.jaspersoft.jaspererver.api.JSEException: mondrian.olap.MondrianException: Mondrian Error:Cube "xxxx": DimensionUsage name ("yyyy") must equal source ("zzzz")	Dimension name yyyy used in MDX query is invalid.

Messages about Loading OLAP Views, continued			
Resource	Error Message	Possible Causes	Resolution
OLAP schema, continued	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: Cannot find shared dimension 'xxxx'	Source of DimensionUsage xxxx is invalid.	Use a valid dimension name for DimensionUsage source. Then reload the schema.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: Failed to parse query 'xxxxxx'	ForeignKey of DimensionUsage in Cube definition is invalid.	Use a valid DimensionUsage foreignKey. Then reload the schema.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: Failed to parse query 'xxxxxx'	Format in measure definition is invalid.	Use a valid format for the measure. Then stop and re-start JasperReports Server, reload the schema.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: Failed to parse query 'xxxxxx'	Column of Measure in Cube definition is invalid.	Use a valid database column for the measure. Then stop and re-start JasperReports Server, and reload the schema.
	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: while parsing catalog null	Aggregator of Measure in Cube definition is invalid.	Use one of { sum, count, min, max, avg, distinct count } as aggregator in the Measure. Then stop and re-start JasperReports Server, and reload the schema.
	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: while parsing catalog null	Data type of Level in Hierarchy definition is invalid.	Use one of { String, Numeric } as data type in the Level. Then stop and re-start JasperReports Server, and reload the schema.
	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: Internal error: while parsing catalog null	Level type of Level in Hierarchy definition is invalid.	Use one of { Regular, TimeYears, TimeQuarters, TimeMonths, TimeWeeks, TimeDays } as level type in the Level. Then stop and re-start JasperReports Server, and reload the schema.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: Failed to parse query 'YYYYYY'	Definition xxxx of dimension hierarchy dddd in cube cccc is incorrect.	Correct the hierarchy definition. Then stop and re-start JasperReports Server, and reload the schema.

Messages about Loading OLAP Views, continued			
Resource	Error Message	Possible Causes	Resolution
OLAP Client connection	An error occurred while getting property "result" from an instance of class com.tonbeller.jpivot.tags.OlapModelProxy	Mondrian schema does not match data source for the Mondrian connection.	Make sure the Mondrian schema is define with the database columns in the selected data source.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: Failed to parse query 'xxxxxx'	The MDX query xxxxxx of the OLAP view does not match with the Mondrian schema with cube yyyy of the OLAP connection.	Make sure the MDX query matches with the Mondrian schema of the OLAP connection for the OLAP view.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: No metadata schema dimensions for catalog: xxxx and cube: yyyy	Catalog xxxx does not match with Mondrian connection in XML/A definitions, or catalog does not match with data source in XML/A connection properties.	Make sure the XML/A catalog matches with Mondrian connection in XML/A definitions, and with the XML/A connection data source.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: Soap Fault code=HTTP fault string=(404)/xxxx/XML/A fault actor=null	XML/A connection URL is invalid.	Make sure the XML/A connection URL, which is derived from JasperReports Server URL, is valid.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: Soap Fault code=HTTP fault string=(401)Bad credentials fault actor=null	XML/A user name and/or password is invalid.	Make sure the XML/A user name and/or password is valid.

Messages about Loading OLAP Views, continued			
Resource	Error Message	Possible Causes	Resolution
Access grant definition	com.jaspersoft.jaspererver.api.JSEException: com.jaspersoft.ji.ja.JasperAnalysisException: MDX object 'xxxx' not found in dimension '[yyyy]'	Invalid or missing level name in a member grant.	Review your AGXML file and correct any incorrect level names you find r add any levels defined in the OLAP schema that are missing from the AGXML. Note: After making this change to the access grant definition, you must edit the Mondrian connection to include the updated AGXML file.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: Error while parsing MDX statement xxxx'	An attribute referred to in the access grant definition does not exist in the profile attribute table.	Review your AGXML file and correct any incorrect attribute names you find. Note: After making this change to the access grant definition, you must edit the Mondrian connection to include the updated AGXML file. If you don't find a problem in the AGXML, the problem may be that the profile attribute really isn't set on the user in question. In this case, you must insert the correct value into the profile attribute table. For more information, refer to the <i>Jaspersoft OLAP User Guide</i> .

3.4.5 Messages that Occur When Expanding Hierarchy Positions and Members

Messages about Expanding Hierarchies			
Resource	Error Message	Possible Causes	Resolution
OLAP schema	An error occurred while getting property "result" from an instance of class com.tonbeller.jpivot.tags.OlapModelProxy	Dimension table name is invalid.	Use a valid dimension table name. Then stop and re-start JasperReports Server, and reload the schema.
	com.jaspersoft.jaspererver.api.JSEException: com.tonbeller.jpivot.olap.model.OlapException: Failed to parse query 'xxxxxx'	Column of Dimension definition is invalid.	Use a valid database column for the dimension. Then stop and re-start JasperReports Server, and reload the schema.
	An error occurred while getting property "result" from an instance of class com.tonbeller.jpivot.tags.OlapModelProxy	Column of Level in dimension hierarchy is invalid.	Use a valid database column for the level. Then stop and re-start JasperReports Server, and reload the schema.

Messages about Expanding Hierarchies, continued			
Resource	Error Message	Possible Causes	Resolution
	An error occurred while getting property "result" from an instance of class com.tonbeller.jpivot.tags.OlapModelProxy	parentColumn of Level in dimension hierarchy is invalid.	Use a valid database column for the level. Then stop and re-start JasperReports Server, and reload the schema.
OLAP schema, continued	An error occurred while getting property "result" from an instance of class com.tonbeller.jpivot.tags.OlapModelProxy	nameColumn of Level in dimension hierarchy is invalid.	Use a valid database column for the level. Then stop and re-start JasperReports Server, and reload the schema.
	An error occurred while getting property "result" from an instance of class com.tonbeller.jpivot.tags.OlapModelProxy	parentColumn of Closure in dimension hierarchy level is invalid.	Use a valid database column for the closure. Then stop and re-start JasperReports Server, and reload the schema.
	An error occurred while getting property "result" from an instance of class com.tonbeller.jpivot.tags.OlapModelProxy	childColumn of Closure in dimension hierarchy level is invalid.	Use a valid database column for the closure. Then stop and re-start JasperReports Server, and reload the schema.
	An error occurred while getting property "result" from an instance of class com.tonbeller.jpivot.tags.OlapModelProxy	Table of Closure in dimension hierarchy level is invalid.	Use a valid database table for the closure. Then stop and re-start JasperReports Server, and reload the schema.
	(Level of dimension hierarchy cannot be expanded)	The nullParentValue of Level in dimension hierarchy is invalid.	Use a valid nullParentValue for the Level. Then stop and re-start JasperReports Server, and reload the schema.

3.4.6 Error Messages that Occur When Drilling Through

Messages about Drilling Through			
Resource	Error Message	Possible Causes	Resolution
OLAP schema	java.sql.SQLException: Unknown column 'xxxx_1.yyyy' in 'where clause'	PrimaryKey yyyy of Hierarchy is invalid.	Use a valid hierarchy primaryKey. Then stop and re-start JasperReports Server, and reload the schema.

CHAPTER 4 SECURING DATA IN JASPERSOFT OLAP



This section describes functionality that can be restricted by the software license for JasperReports Server. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

When a particular set of data is accessed by different people, you may need to restrict the data that is displayed to each one. For example, you might allow employees to see only their own personnel data, supervisors to see selected portions of the data of the workers they supervise, and human resources staff to see everyone's complete data.

Jaspersoft OLAP commercial editions' data security enables administrators to define data access for users in a flexible, scalable way. When properly configured, a user only sees the data that the company wants him to see. To define this access, add data access filtering rules (grants) to the Mondrian connections defined in Jaspersoft OLAP. When combined with user profile data, these rules are powerful and flexible.

The power of this solution is best presented through a sample business case. This section describes a fictional company's implementation from both a business perspective and an implementation perspective.

For details about the basics of Jaspersoft OLAP, refer to the user guide for your edition of Jaspersoft OLAP; it is included in the product distribution.

This chapter has these sections:

- **Securing Jaspersoft OLAP Data: A Business Case**
- **Overview of CZS's Process**
- **Understanding Access Grant Definitions and Profile Attributes**
- **Configuring CZS's OLAP view**
- **Reference Material**



This chapter assumes that you have the administrator role. It describes a number of tasks that only administrators can perform. The sections about the profile attribute table also assume database and SQL knowledge.

4.1 Securing Jaspersoft OLAP Data: A Business Case

CZS is an up-and-coming consumer electronics company with operations in the U.S. and Japan. CZS uses Jaspersoft OLAP to track sales data, such as sales revenue and operating cost.

CZS employs the following sales staff:

- Rita is the regional sales manager in the Western U.S. She uses the Sales Numbers OLAP view to track sales trends in her region.

- Pete is a Sales Rep selling televisions in Northern California. He uses the same view to track his quarterly progress.
- Yasmin is a Sales Rep selling cell phones in Northern California. She uses the same view to track her quarterly progress.
- Alexi is the regional sales manager in Kansai, Japan. He uses the same view to track sales trends in his region.

CZS stores its data in MySQL database tables. The data is exposed by the Sales Numbers OLAP view, which displays information about CZS’s consumer electronics sales across the world. It is filtered according to each user’s role, geographical area, and product.

This chapter describes how CZS addressed their business case using Jaspersoft OLAP. It describes the specific steps CZS took to secure their data for users with certain roles and profile attributes.

4.2 Overview of CZS’s Process

After articulating their business case (as above), CZS took these steps to implement the Sales Numbers view:

Step		Described in Section ...
1	Identified the dimensions upon which to base access control. CZS chose the Geographical Area and Product Department dimensions.	4.4.1.2, “Dimensions,” on page 67
2	Identified and created access roles. CZS identified two roles: one for managers and another for sales reps. Both are granted access to the OLAP view.	4.4.3, “Determining Roles,” on page 68
3	Assigned the appropriate roles to each user based on each employees’ responsibilities.	4.4.4, “Selecting Users for the Roles,” on page 69
4	Identified the attributes that will be defined in the profile attribute table. CZS identified five attributes: Country, Region, State, Cities, and ProductDepartment.	4.3.2, “Profile Attributes and Variable Substitution,” on page 67 and 4.4.5, “Assigning Profile Attributes,” on page 70
5	Inserted the correct values for each user into the profile attribute table.	4.4.5.1, “Defining Profile Attributes for Users,” on page 71 and the <i>JasperReports Server Ultimate Guide</i> .
6	Created an AGXML (access grant definition XML) file that defines the access granted to users with each role and profile attribute.	4.4.5.2, “Using Profile Attributes in Access Grant Definition,” on page 71 4.5.2, “Access Grant Definition,” on page 75
7	Created a Mondrian connection that pointed to their sales data and included the sales access grant definition.	4.4.6, “Creating a Mondrian Connection,” on page 71
8	Created the OLAP view that points to the Mondrian connection.	4.4.7, “Generating a Sales OLAP View,” on page 72
9	Tested the view as various users.	4.4.8, “Testing the Results,” on page 72

4.3 Understanding Access Grant Definitions and Profile Attributes



The examples in this section are meant to illustrate the basic concepts of an access grant definition. They are not realistic examples, nor do they represent best practices for data security. Instead, they are meant to introduce the various parts of an access grant definition. The last section in this chapter is a more realistic model; see [4.5.2, “Access Grant Definition,” on page 75](#).

4.3.1 Access Grant Definitions

An access grant definition is an XML file that specifies each role’s access rights to various parts of the data defined in a cube. It refers to access roles defined in JasperReports Server as well as levels in dimensions defined in a particular cube. An access grant definition can control access at any level of your dimensions, and like an OLAP schema, it follows the Mondrian XML format.

CZS started by securing their data along the Geographical Area dimension of the cube. A simple access grant for this dimension might be similar to the following:

```
<MemberGrant member="[Geographic Area].[USA].[West].[CA]" access="all"/>
```

This member grant gives the role in question full access to the CA (California) member of the dimension. California is at the state level of the dimension; you could just as easily grant access to all of the West region or to individual cities.

The following code fragment shows the above member grant in the context of a full role definition. It grants the ROLE_CA role access to all California data:

```
<Roles>
  <Role name="ROLE_CA">
    <SchemaGrant access="none">
      <CubeGrant cube="Sales" access="all">
        <HierarchyGrant hierarchy="[Geographical Area]" access="custom"
          topLevel="[Geographical Area].[State]"
          bottomLevel="[Geographical Area].[City]">
          <MemberGrant member="[Geographical Area].[USA].[Western].[CA]"
            access="all"/>
        </CubeGrant>
      </SchemaGrant>
    </Role>
  </Roles>
```

Returning to the business case requirements, CZS also wants to grant access based on the Product dimension. To do so, they could add a hierarchy grant like the following one to the cube grant:

```
<Roles>
  <Role name="ROLE_CA">
    <SchemaGrant access="none">
      <CubeGrant cube="Sales" access="all">
        <HierarchyGrant hierarchy="[Geographical Area]" access="custom"
          topLevel="[Geographical Area].[State]"
          bottomLevel="[Geographical Area].[City]">
          <MemberGrant member="[Geographical Area].[USA].[Western].[CA]"
            access="all"/>
        </HierarchyGrant>
        <HierarchyGrant hierarchy="[Product]" access="custom"
          topLevel="[Product].[Product Family]"
          bottomLevel="[Product].[Product Department]">
        </HierarchyGrant>
      </CubeGrant>
    </SchemaGrant>
  </Role>
</Roles>
```

```

        <MemberGrant member="[Product].[Electronics].[WirelessDevices]"
            access="all" />
    </HierarchyGrant>
</CubeGrant>
</SchemaGrant>
</Role>
</Roles>

```

Additionally, CZS could use an AGXML file to define access for multiple roles, as in the following example, which grants access to a second role called `ROLE_SF`:

```

<Roles>
  <Role name="ROLE_CA">
    <SchemaGrant access="none">
      <CubeGrant cube="Sales" access="all">
        <HierarchyGrant hierarchy="[Geographical Area]" access="custom"
            topLevel="[Geographical Area].[State]"
            bottomLevel="[Geographical Area].[City]">
          <MemberGrant member="[Geographical Area].[USA].[Western].[CA]"
              access="all" />
        </HierarchyGrant>

        <HierarchyGrant hierarchy="[Product]" access="custom"
            topLevel="[Product].[Product Family]"
            bottomLevel="[Product].[Product Department]">
          <MemberGrant member="[Product].[Electronics].[Wireless Devices]"
              access="all" />
        </HierarchyGrant>
      </CubeGrant>
    </SchemaGrant>
  </Role>

  <Role name="ROLE_SF">
    <SchemaGrant access="none">
      <CubeGrant cube="Sales" access="all">
        <HierarchyGrant hierarchy="[Geographical Area]" access="custom"
            topLevel="[Geographical Area].[City]"
            bottomLevel="[Geographical Area].[City]">
          <MemberGrant member="[Geographical Area].[USA].[Western].[CA].
              [San Francisco]" access="all" />
        </HierarchyGrant>

        <HierarchyGrant hierarchy="[Product]" access="custom"
            topLevel="[Product].[Product Family]"
            bottomLevel="[Product].[Product Department]">
          <MemberGrant member="[Product].[Electronics].[WirelessDevices]"
              access="all" />
        </HierarchyGrant>
      </CubeGrant>
    </SchemaGrant>
  </Role>
</Roles>

```

However, the complexity of a complete implementation structured as shown would require reams of XML and an access role configuration that took into account geographical location. Therefore, to avoid this complexity, CZS defined profile attributes that describe each user in terms of geography and product. Then, using variables to represent the attributes, they wrote access grant definitions, as described in the following sections.

4.3.2 Profile Attributes and Variable Substitution

A profile attribute is a name-value pair defined at the user or role level; in this example, it is set on the user and corresponds to a member of a dimension. Jaspersoft OLAP uses these attributes to determine the access to grant to a user based on her role. Variable substitution in the access grant definition can refer to these attributes when determining access.

For example, take Rita and Alexi; both have the same role and the same access to the Sales Numbers OLAP view, but CZS doesn't want them to see the same data—Rita should see data about California and Alexi should see data about Osaka. Without profile attributes, this would only be possible if CZS's access roles were defined along geographical lines. Instead, CZS defines profile attributes that indicate each user's geographical affiliations: Rita is responsible for California and Alexi is responsible for Osaka.

If we were only interested in the City level of the dimension, an access grant that used variable substitution would be similar to the following:

```
<MemberGrant member="[Geographic Area].[USA].[West].[CA].[%{Cities}]" access="all"/>
```

This grant gives specific access to USA, West, and California, but uses variable substitution (represented by [%{Cities}]) to grant access to whichever cities are defined in the profile attribute table for the specific user.

CZS also defined profile attributes for other levels of the Geographical Area dimension. A member grant that searched all of these attributes would be similar to the following:

```
<MemberGrant member="[Geographical Area].[%{Country}].[%{Region}].[%{State}].[%{Cities}]" access="all"/>
```

The access grant definition that CZS created, including the profile attributes, is shown in [4.5.2, “Access Grant Definition,” on page 75](#).

4.4 Configuring CZS's OLAP view

The following sections go into more depth about how to define and take advantage of profile attributes.

4.4.1 Defining a Sales Numbers Cube

The following sections describe the measures and dimensions in the cube that represent CZS's sales data.

4.4.1.1 Measures

CZS is primarily interested in the volume and revenue of their sales, as well as their operational cost. These metrics are represented in Jaspersoft OLAP as measures defined in a cube. The measures can be aggregated (rolled up) by dimension hierarchies (described below), such as product, geographical area, and time.

CZS's measures are numeric values contained in the sales_fact_2006 fact table. This fact table includes three measures: unit sales, store cost, and store sales.

4.4.1.2 Dimensions

Dimensions are categorical entities used to analyze measures. The cube underlying the Sales Numbers OLAP view includes two dimensions:

- The geographical dimension includes these levels: country, region, state, and city.

- The product dimension describes the merchandise being sold and includes these levels: product family, product line, and product.

For more information about how the Sales Numbers schema joins the hierarchical tables with the detail tables, refer to [4.5.1, “OLAP Schema,” on page 74](#).

4.4.2 Creating an OLAP Schema

Having defined their dimensions and measures, CZS created the *czs-sales* OLAP schema. An OLAP schema is an XML file that defines a cube’s structure. The Jaspersoft OLAP Workbench can simplify the task of creating a schema. For details, refer to the documentation provided with the Workbench.

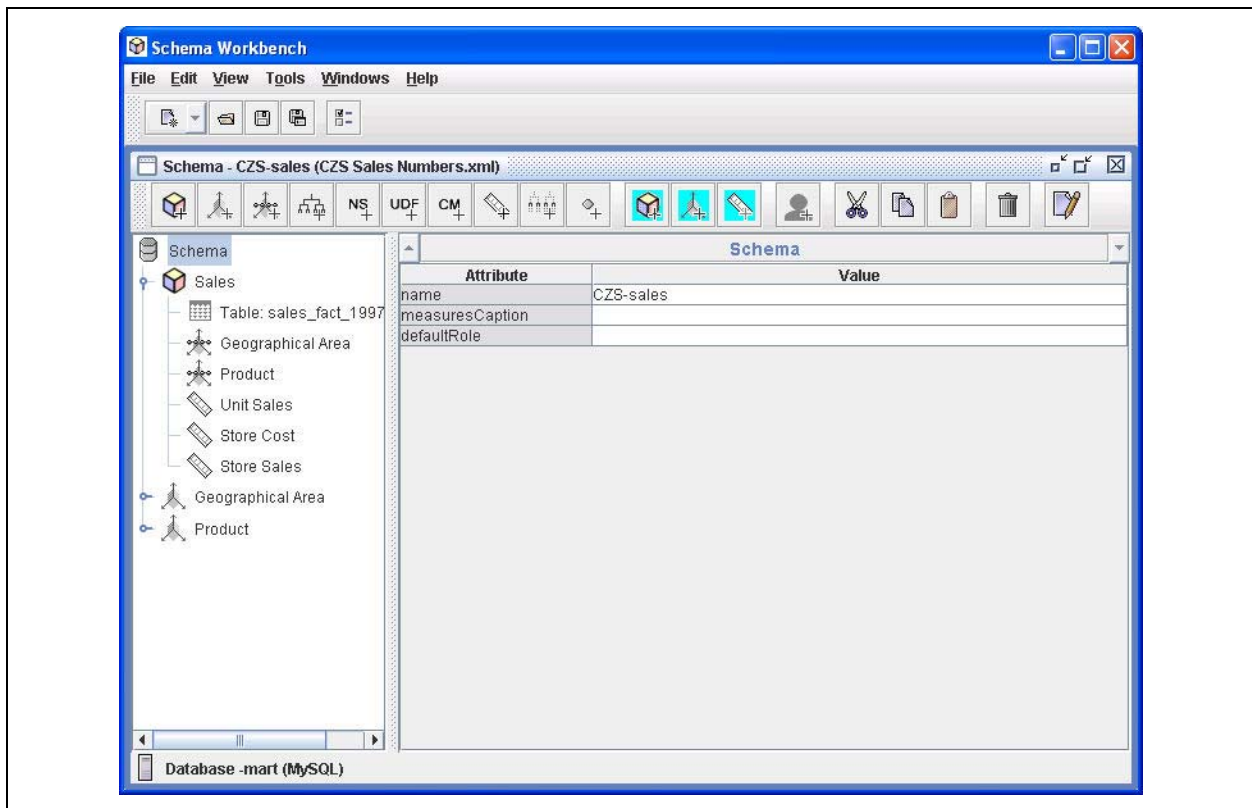


Figure 4-1 Jaspersoft OLAP Workbench

Once the OLAP schema is created, add it to the repository. For details, refer to the *Jaspersoft OLAP User Guide*.

For the XML behind this OLAP schema, refer to [4.5.1, “OLAP Schema,” on page 74](#).

4.4.3 Determining Roles

Data security in Jaspersoft OLAP relies on a user’s roles to determine the access permissions to grant. CZS defined these roles:

- `ROLE_SALES_MANAGER` is assigned to Sales Managers.
- `ROLE_SALES_REP` is assigned to Sales Reps.

CZS grants each role sufficient access to view the Sales Numbers OLAP view. For details about creating roles and assigning privileges, refer to the *JasperReports Server Administrator Guide*.

The following shows CZS's ROLE_SALES_MANAGER:

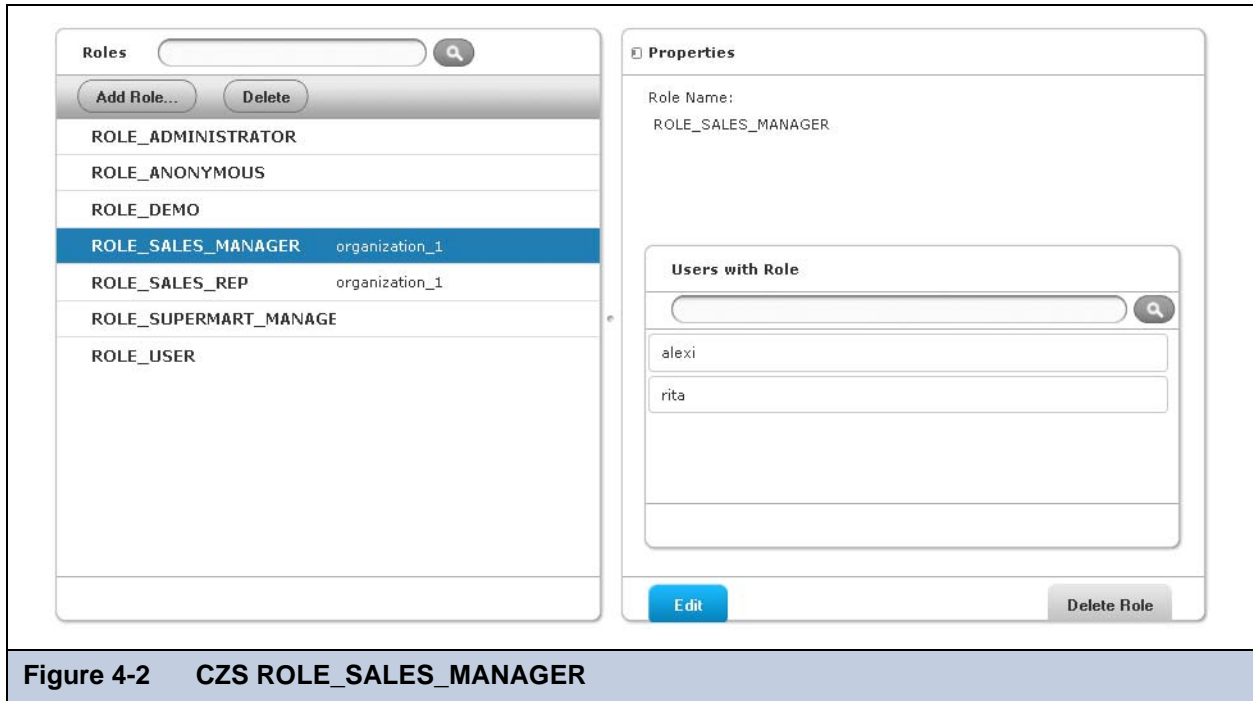


Figure 4-2 CZS ROLE_SALES_MANAGER

4.4.4 Selecting Users for the Roles

CZS defined a username for each of their employees. It then assigned roles to each employee; the roles are consistent with the employees' levels of responsibility. For details about creating users, refer to the *JasperReports Server Administrator Guide*.

Rita	ROLE_SALES_MANAGER
Pete	ROLE_SALES_REP
Yasmin	ROLE_SALES_REP
Alexi	ROLE_SALES_MANAGER

The following figure shows the configuration of Rita's user account:

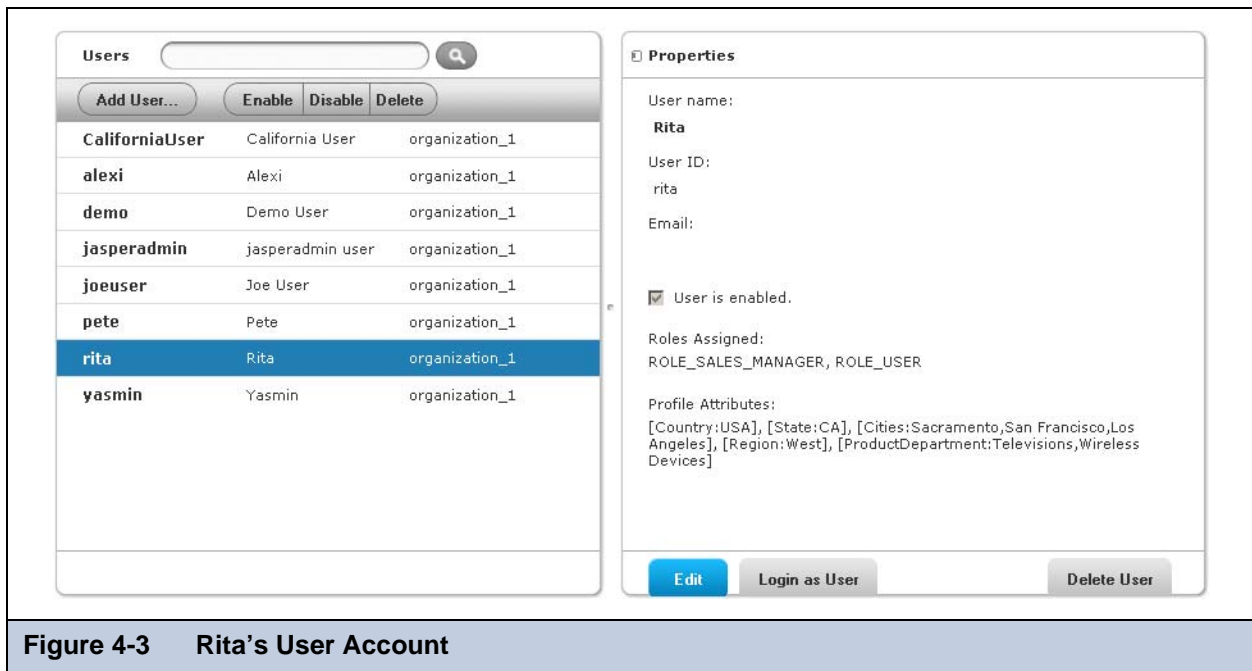


Figure 4-3 Rita’s User Account

4.4.5 Assigning Profile Attributes



Profile attributes are defined in a table in the JasperServer repository database. To update this table, you must use database tools, such as the MySQL command line, or custom code. See *JasperReports Server Ultimate Guide* for more information about defining the attributes.

Notice the profile attributes listed below the user’s roles in **Figure 4-3**. CZS implemented them by taking advantage of the variable substitution feature, which simplifies the creation of the access grant definition. This section explains the concept.

CZS determined that they need to describe their users in terms of the product lines that they sell and the geographical areas where they sell. Thus, each CZS user is assigned five profile attributes:

- Four describe the employee’s geographical area of responsibility: country, region, state, and city. These attributes correspond to the levels of the Geographical Area dimension. The access grant definition shown in **4.3.2, “Profile Attributes and Variable Substitution,” on page 67** refers to these profile attributes.
- One describes the products the employee is responsible for selling: product department. This attribute corresponds to the Product Line level of the product dimension.

Each user’s attributes determine the data returned to him by the view, based on an access grant definition that refers to profile attributes by using variable substitution. For example, Rita’s attribute value for Cities is San Francisco, Los Angeles, Sacramento while Pete’s is San Francisco. Thus, Pete sees a subset of the data Rita sees.

CZS assigned the following attributes to their users:

Attributes of CZS Users					
User	Profile Attributes				
	Geographical				Product/Department
	Country	Region	State	Cities	
Rita	USA	West	CA	San Francisco, Los Angeles, Sacramento,	Television, Wireless Devices
Pete	USA	West	CA	San Francisco	Television
Yasmin	USA	West	CA	San Francisco	Wireless Devices
Alexi	Japan	Kansai	Osaka	Osaka, Sakai	Wireless Devices

At a high level, CZS took the following steps in populating and utilizing the profile attribute table:

1. Retrieved each user ID by querying the JasperReports Server database. This information is necessary to identify the user when adding or updating a profile attribute.
2. Inserted the required profile attributes into the table for each user. This defines the profile attributes referred to by variable substitution in the AGXML file.
3. Created an access grant definition (AGXML file) that refers to these attributes.
4. Used the access grant definition in a Mondrian connection.
5. Used the Mondrian connection in an OLAP view.

For the necessary details about configuring profile attributes and Mondrian connections, refer to:

- [4.4.5.1, “Defining Profile Attributes for Users,” on page 71](#)
- [4.4.5.2, “Using Profile Attributes in Access Grant Definition,” on page 71](#)
- [4.4.6, “Creating a Mondrian Connection,” on page 71](#)
- [4.4.7, “Generating a Sales OLAP View,” on page 72.](#)

4.4.5.1 Defining Profile Attributes for Users

Profile attributes are defined in a table in the repository database. To update this table, you must use database tools, such as the MySQL command line. For detailed steps, refer to the *JasperReports Server Ultimate Guide*.

4.4.5.2 Using Profile Attributes in Access Grant Definition

With the profile attributes defined, CZS took advantage of them using variable substitution in their access grant definitions, as described in [4.3.2, “Profile Attributes and Variable Substitution,” on page 67](#). For specific information about the format of these variables, refer to the *Jaspersoft OLAP User Guide*.

CZS’s access grant definition (czs-access-grant.agxml) is shown in [4.5.2, “Access Grant Definition,” on page 75](#). It utilizes the profile attributes discussed in this section (Country, Region, State, Cities, and ProductDepartment).

4.4.6 Creating a Mondrian Connection

CZS created a Mondrian connection that included the czs-sales-grant.agxml access grant definition file. OLAP views and XML/A connections that use this Mondrian connection impose the defined data level security rules.

For instructions on creating a Mondrian connection that includes an access grant, refer to the *Jaspersoft OLAP User Guide*.

4.4.7 Generating a Sales OLAP View

Having created the profile attributes, access grant definition, and a Mondrian connection, CZS then created the actual Sales Numbers OLAP view to run against the Mondrian connection. This OLAP view pointed to the Mondrian connection that includes the `czs-sales-grant.agxml` access grant definition file. The MDX query for this view is:

```
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]}
on columns, {[Geographical Area], [Product]} ON rows from Sales
```

Once the OLAP view was created, CZS updated the repository object permissions so that each access role (ROLE_SALES_MANAGER and ROLE_SALES_REP) had Read access to the view. Read access is required to open OLAP views.

For instructions on access control, refer to the administrator guide for your edition of JasperReports Server, and to the *JasperReports Server Ultimate Guide*. For information about creating OLAP views, refer to the *Jaspersoft OLAP User Guide*.

4.4.8 Testing the Results

Finally, CZS tested the view security by logging in as each CZS user and checking the user's access to the view.

To test the access granted to users on data in the CZS OLAP view:

1. Click **Manage > Users**.
2. In the Users panel, select the CZS user to test.
3. In the user's Properties panel, click **Login as User**.
The selected user's Home page appears.
4. Click **View > OLAP Views**.
5. In the Search panel, click **CZS Sales Numbers OLAP View**.
The view appears.
6. Test the view to ensure that it only displays data the user should see and does not hide other data the user should see but cannot.
7. Click **Log Out** to return to your Home page, then login as each of the other users.

When reviewing the CZS OLAP views in the following figures (figures 4-4 to 4-7), note the different access each user has to the analysis:

- Rita can see all data for the state of California and the three California cities where CZS has offices (Los Angeles, Sacramento, and San Francisco):

Dimensions				Measures		
State	City	Product Family	Product Department	Unit Sales	Store Cost	Store Sales
CA		Electronics		60,758	51,642.62	129,396.41
		Electronics	Televisions	7,102	5,662.27	14,203.24
			Wireless Devices	53,656	45,980.35	115,193.17
CA		Electronics		60,758	51,642.62	129,396.41
		Electronics	Televisions	7,102	5,662.27	14,203.24
			Wireless Devices	53,656	45,980.35	115,193.17
CA	Los Angeles	Electronics		20,929	17,692.58	44,370.84
		Electronics	Televisions	2,560	2,014.67	5,065.10
			Wireless Devices	18,369	15,677.91	39,305.74
	Sacramento	Electronics		20,716	17,542.74	44,011.34
		Electronics	Televisions	2,422	1,953.55	4,823.88
			Wireless Devices	18,294	15,589.18	39,187.46
	San Francisco	Electronics		19,113	16,407.31	41,014.23
		Electronics	Televisions	2,120	1,694.05	4,314.26
			Wireless Devices	16,993	14,713.26	36,699.97

Filter:

Figure 4-4 Rita's View

- Pete can only see television data about San Francisco:

Dimensions		Measures			
City	Product Department	Unit Sales	Store Cost	Store Sales	
San Francisco	Televisions	2,120	1,694.05	4,314.26	

Filter:

Figure 4-5 Pete's View

- Yasmin can only see wireless devices data about San Francisco:

Dimensions		Measures			
City	Product Department	Unit Sales	Store Cost	Store Sales	
San Francisco	Wireless Devices	16,993	14,713.26	36,699.97	

Filter:

Figure 4-6 Yasmin's View

- Alexi can only see data for Osaka prefecture and the two cities there where CZS has offices (Osaka and Sakai):

Dimensions				Measures		
State	City	Product Family	Product Department	Unit Sales	Store Cost	Store Sales
Osaka		Electronics		54,643	45,803.80	114,701.96
		Electronics	Wireless Devices	48,537	40,967.45	102,564.67
Osaka	Osaka	Electronics		18,632	15,800.79	39,619.66
	Sakai	Electronics		29,905	25,166.66	62,945.01

Filter:

Figure 4-7 Alexi's View

4.5 Reference Material

4.5.1 OLAP Schema

The CZS-sales.xml OLAP schema defines a cube that is based on the sales_fact_2006 table. It uses three measures: Unit Sales, Store Cost, and Store Sales. The cube can be analyzed with its two dimensions: Geographical Area and Product.

```
<Schema name="CZS-sales">
  <Dimension name="Geographical Area">
    <Hierarchy hasAll="true" primaryKey="store_id" primaryKeyTable="store">
      <Join leftKey="region_id" rightKey="region_id">
        <Table name="store" />
        <Table name="region" />
      </Join>
      <Level name="Country" table="region" column="sales_country" type="String"
        uniqueMembers="true" levelType="Regular" hideMemberIf="Never" />
      <Level name="Region" table="region" column="sales_region" type="String"
        uniqueMembers="false" levelType="Regular" hideMemberIf="Never" />
      <Level name="State" table="store" column="store_state" type="String"
        uniqueMembers="true" levelType="Regular" hideMemberIf="Never" />
      <Level name="City" table="store" column="store_city" type="String"
        uniqueMembers="false" levelType="Regular" hideMemberIf="Never" />
      <Level name="Store Name" table="store" column="store_name" type="String"
        uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        <Property name="Store Type" column="store_type" type="String" />
        <Property name="Store Manager" column="store_manager" type="String" />
        <Property name="Store Sqft" column="store_sqft" type="Numeric" />
        <Property name="Street address" column="store_street_address"
          type="String" />
      </Level>
    </Hierarchy>
  </Dimension>
  <Dimension name="Product">
    <Hierarchy hasAll="true" primaryKey="product_id" primaryKeyTable="product">
```

```

<Join leftKey="product_class_id" rightKey="product_class_id">
  <Table name="product" />
  <Table name="product_class" />
</Join>
<Level name="Product Family" table="product_class" column="product_family"
  type="String" uniqueMembers="true" levelType="Regular" hideMemberIf="Never" />
<Level name="Product Department" table="product_class"
  column="product_department" type="String" uniqueMembers="false"
  levelType="Regular" hideMemberIf="Never" />
<Level name="Product Category" table="product_class" column="product_category"
  type="String" uniqueMembers="false" levelType="Regular" hideMemberIf="Never" />
<Level name="Product Subcategory" table="product_class"
  column="product_subcategory" type="String" uniqueMembers="false"
  levelType="Regular" hideMemberIf="Never" />
<Level name="Brand Name" table="product" column="brand_name" type="String"
  uniqueMembers="false" levelType="Regular" hideMemberIf="Never" />
<Level name="Product Name" table="product" column="product_name" type="String"
  uniqueMembers="true" levelType="Regular" hideMemberIf="Never" />
</Hierarchy>
</Dimension>
<Cube name="Sales" cache="true" enabled="true">
  <Table name="sales_fact_2006" />
  <DimensionUsage source="Geographical Area" name="Geographical Area"
    foreignKey="store_id" />
  <DimensionUsage source="Product" name="Product" foreignKey="product_id" />
  <Measure name="Unit Sales" column="unit_sales" formatString="Standard"
    aggregator="sum" />
  <Measure name="Store Cost" column="store_cost" formatString="#,###.00"
    aggregator="sum" />
  <Measure name="Store Sales" column="store_sales" formatString="#,###.00"
    aggregator="sum" />
</Cube>
</Schema>

```

4.5.2 Access Grant Definition

The CZS-sales-grant.agxml access grant definition is modeled after the CZS-sales.xml OLAP schema and defines access for users with the ROLE_SALES_MANGER and ROLE_SALES_REP access roles. It uses variable substitution to refer to profile attributes that determine the data the user can see in the OLAP view.

```

<Roles>
  <Role name="ROLE_SALES_MANAGER">
    <SchemaGrant access="none">
      <CubeGrant cube="Sales" access="all">
        <HierarchyGrant hierarchy="[Geographical Area]" access="custom"
          topLevel="[Geographical Area].[State]"
          bottomLevel="[Geographical Area].[City]">
          <MemberGrant member="[Geographical
            Area].[#{Country}].[#{Region}].[#{State}]" access="all"/>
        </HierarchyGrant>
      </CubeGrant>
    </SchemaGrant>
  </Role>
</Roles>

```

```

    <HierarchyGrant hierarchy="[Product]" access="custom"
      topLevel="[Product].[Product Family]"
      bottomLevel="[Product].[Product Department]">
      <MemberGrant member="[Product].[Electronics].[#{ProductDepartment}]"
        access="all"/>
    </HierarchyGrant>
  </CubeGrant>
</SchemaGrant>
</Role>
<Role name="ROLE_SALES_REP">
  <SchemaGrant access="none">
    <CubeGrant cube="Sales" access="all">
      <HierarchyGrant hierarchy="[Geographical Area]" access="custom"
        topLevel="[Geographical Area].[City]"
        bottomLevel="[Geographical Area].[City]">
        <MemberGrant member="[Geographical
          Area].[#{Country}].[#{Region}].[#{State}].[#{Cities}]" access="all"/>
      </HierarchyGrant>

      <HierarchyGrant hierarchy="[Product]" access="custom"
        topLevel="[Product].[Product Family]"
        bottomLevel="[Product].[Product Department]">
        <MemberGrant member="[Product].[Electronics].[#{ProductDepartment}]"
          access="all"/>
      </HierarchyGrant>
    </CubeGrant>
  </SchemaGrant>
</Role>
</Roles>

```

CHAPTER 5 ADMINISTERING JASPERSOFT OLAP

This chapter provides guidance for JasperSoft OLAP users with administrative responsibilities. It assumes that such users have extensive technical knowledge of databases, system integration, and multidimensional modeling.

The chapter has these sections:

- **Understanding the Design Concepts**
- **Maintaining the System**
- **Tuning Performance**
- **Integrating JasperSoft OLAP in the Enterprise's Data Flow**
- **Troubleshooting Information for Technical Support**

5.1 Understanding the Design Concepts

Designing a schema requires an understanding of dimensional modeling with cubes, dimensions, and measures.

5.1.1 Introduction

Dimensional modeling is a way of organizing information so that data is better suited to certain types of queries. Such queries often arise during statistical analysis of numerical trends and patterns. In dimensional modeling, cubes, dimensions, and measures constitute a logical schema that describes your business.

Dimensional modeling typically differentiates analysis from other types of reporting:

- Transactional reports typically display data in a fixed format; the structure is always the same, but the quantitative and qualitative information for each time period or subject of study differ.
- Analysis typically provides access to large amounts of data aggregated with respect to different variables. Each query may produce a view with a different structure. Analysis is generally interactive: it answers questions, helps you find trends and outliers in your data, and get a deeper understanding of your data.

Another way analytical reporting differs from traditional reporting is its emphasis on numerical data. While an event can be described qualitatively (“Joe and Sue were married on June 15th in the park by a minister”), these details could be better described if you want to summarize and examine information about a large number of marriages. In this case, you’d be better off with data organized dimensionally.

Consider the five basic questions journalists ask: who, what, when, where, and why?

These questions are helpful in organizing data. Defining a standardized way to describe groups of events makes it easier to pose certain queries. For example:

- How many marriages occurred in parks in June versus in December?
- What percentage of marriages performed by ministers occurred in a church?

If the data is organized dimensionally, we can answer such questions with a single query, rather than by examining each qualitative wedding announcement in the paper.

5.1.2 Designing the Model

Facilitating questions and answers about large sets of data involves some planning and intuition to know which questions are relevant.

Even if you don't know how a particular variable affects the results, you may still want to model it if you suspect an influence; this may help you to understand the variable's effects.

The basic premise is that we can lay our data out such that mathematics can reveal meaning. It is essential to phrase measures quantitatively. For example, instead of posing a yes or no question, provide a count of the each possible answer, either 1 or 0. This way, the number of yes answers can be tallied at any level of aggregation and for any set of dimensional parameters.

In an OLAP schema:

- A cube is the logical entity that holds all the facts for each measure. Cubes can contain millions of facts, which are too numerous to be analyzed individually.
- A fact is the value of a measure for a specific member of a dimension; facts relate directly to their dimensional positions.
- Dimensions provide the structure for slicing and dicing data. Each fact references the dimensional members that correspond to the variables defining the observation of this fact. Generally speaking, the most useful way to aggregate data reflects the domain model; for companies, the data organization should reflect their business rules.

The art of dimensional modeling lies in finding useful ways to aggregate your data.

Dimensions are organized into hierarchies that represent increasing levels of aggregation. Consider the example of a dimension representing time. Such dimensions often include levels representing month, day, and year. If your requirements necessitate it, the dimension might also include levels for quarters, fiscal quarters, 10-day periods, weeks, or even days.

The hierarchical nature of dimensions allows both roll-up and drill-down. For example, if fiscal quarters are made up of 3 months, the quarterly data can be broken out by month. Then, if the measures for a particular month are anomalous, you can drill deeper into that month (grouping results by day) to identify the cause of the anomaly. You can also compare your expectations and actual results in the same OLAP view by representing them both as measures.

Measures are essentially numerical formulas. The values they resolve to are called facts. For example, if weight is the measure, 10 pounds is a fact. To be useful, a fact needs context, such as what object weighs 10 pounds and when it weighed 10 pounds. In a business context, common measures may include number of units, transactions, cost per unit, and price per unit. Calculated measures allow for more complex formulas that work off of other measures. So, for example, operating profit margin may be a calculated measure that is a formula based on many measures.

Restating briefly, cubes contain facts, which are instances of measures; these facts are defined by dimensional coordinates.

For businesses, useful dimensions may include time, locations, product categorizations, customers, employees, and any other categorizations that provide business context for individual records. Organizing each dimension into a hierarchy allows roll-up and drill-down.

Designing and using a dimensional model involves creating processes, such as the following:

Creating a Dimensional Model	
Process in Dimensional Modeling	Examples of Process Applied to Specific Data
Phrasing the business problems to be understood and solved.	<ul style="list-style-type: none"> ♦ How does customer satisfaction affect revenue? ♦ How can we improve customer satisfaction?
Identifying quantifiable measures of interest, Key Performance Indicators (KPIs), and metrics.	<ul style="list-style-type: none"> ♦ Revenue ♦ Customer survey results ♦ Repeat customers
Identifying variables that are thought to influence those measures.	<ul style="list-style-type: none"> ♦ Interactions between the customer and business ♦ Quality of the product or service ♦ Preferences of individual customers.
Organizing those variables into dimensional hierarchies.	<ul style="list-style-type: none"> ♦ Product dimension with levels for Line, Category, Product, and SKU ♦ Store Location dimension with levels for City, State, Country
Expressing those measures and variables in cubes and dimensions.	<ul style="list-style-type: none"> ♦ Dimensions - Store location, Products, Time, Customers, CRM cases ♦ Cubes - Sales, Accounts, Surveys ♦ Measures - Satisfaction Score, Number of Interactions, Number of Transactions, Prices, Size of Transactions, Revenues
Mapping the source data into a schema for analysis.	<ul style="list-style-type: none"> ♦ SugarCRM source data --> SugarCRM-ops.xml
Making queries whose answers provide analytical insight.	<ul style="list-style-type: none"> ♦ What are the average Satisfaction Scores and Size of Transactions for each store by month? ♦ Why does the September Revenue appear to be an outlier for stores in Florida, although the Satisfaction Scores are within the normal range? ♦ Why are the Satisfaction Scores and Number of Repeat Transactions consistently lower for one particular store than the average across the business?

5.1.3 Applying the Model

5.1.3.1 Example Using FoodMart Data

This example is based on the FoodMart sample data that can be installed with JasperReports Server. It illustrates the types of questions that can be answered interactively using Jaspersoft OLAP.

Each step in this example shows the corresponding MDX query so that you can explore the data on your own.

To analyze the FoodMart view:

1. We'll begin by looking at a top-level summary of sales for all products. An aggregated view like this lets us dive into the data in any direction of interest. In this example, we look at data for the month of December.


Dimensions		Measures
Product		Store Sales
All Products		56,965.64

Filter: Month=[Time].[2006].[Q4].[12]

Figure 5-1 Top-level Summary for December

```
select {[Measures].[Store Sales]} ON COLUMNS,
Hierarchize({[Product].[All Products]}) ON ROWS
from [Sales]
where [Time].[2006].[Q4].[12]
```

2. Drill-down is the most basic function in an analysis.


To drill-down, click  next to the ALL PRODUCTS member; the next level of the PRODUCT hierarchy appears.

Dimensions			Measures
Product	Product Family		Store Sales
All Products			56,965.64
All Products	Drink		4,802.03
	Food		41,484.40
	Non-Consumable		10,679.21

Filter: Month=[Time].[2006].[Q4].[12]

Figure 5-2 Expand All Products

```
select {[Measures].[Store Sales]} ON COLUMNS,
  Hierarchize(Union({[Product].[All Products]}, [Product].[All Products].Children))
ON ROWS
from [Sales]
where [Time].[2006].[Q4].[12]
```

3. Drill-down again, this time by zooming on the Drink product family. Use  on ALL PRODUCTS then DRINK so that the outermost left-hand column is PRODUCT FAMILY instead of the highest level category in this dimension (ALL PRODUCTS).

Dimensions			Measures
Product	Product Family	Product Department	Store Sales
All Products	Drink	Alcoholic Beverages	1,441.48
		Beverages	2,655.45
		Dairy	705.10

Filter: Month=[Time].[2006].[Q4].[12]

Figure 5-3 Zoom on the Drink Product Family

```
select {[Measures].[Store Sales]} ON COLUMNS,
  [Product].[Drink].Children ON ROWS
from [Sales]
where [Time].[2006].[Q4].[12]
```


4. By zooming, we could drill-down to the lowest level of data in the cube. Let's instead focus our attention by zooming on the ALCOHOLIC BEVERAGES product department, then the BEER AND WINE product category.

Dimensions					Measures
Product	Product Family	Product Department	Product Category	Product Subcategory	Store Sales
All Products	Drink	Alcoholic Beverages	Beer and Wine	Beer	361.92
				Wine	1,079.56


Filter: Month=[Time].[2006].[Q4].[12]

Figure 5-4 Expand Beer and Wine Subcategories

```
select {[Measures].[Store Sales]} ON COLUMNS,
       [Product].[Drink].[Alcoholic Beverages].[Beer and Wine].Children ON ROWS
from [Sales]
where [Time].[2006].[Q4].[12]
```

Now we can see that, overall, wine makes up the majority of the dollar amount for this category.

5. Let's see if that trend holds up across different stores. To do so, we add another dimension, which creates a crossjoin. Analytic views make creating a crossjoin easy.

- a. Select  and make STORE a row and select **All Stores** (for instructions on adding a dimension, refer to [3.3.2, "Cube Configuration," on page 40](#)). Adding the STORE dimension to rows allows you to see the breakdown within the store as well as across different stores with respect to certain products (if we wanted to compare different products across stores, we would make PRODUCTS a row instead of STORES).

Dimensions					Measures	
Product	Product Family	Product Department	Product Category	Product Subcategory	Store	Store Sales
All Products	Drink	Alcoholic Beverages	Beer and Wine	Beer	All Stores	361.92
				Wine	All Stores	1,079.56

Filter: Month=[Time].[2006].[Q4].[12]

Figure 5-5 Adding the All Stores Dimension

```
select {[Measures].[Store Sales]} ON COLUMNS,
       Crossjoin([Product].[Drink].[Alcoholic Beverages].[Beer and Wine].Children,
                {[Store].[All Stores]}) ON ROWS
from [Sales]
where [Time].[2006].[Q4].[12]
```

- b. Then drill-down to navigate to intersections of the store and city data. For example, drill-down on ALL STORES then CALIFORNIA to examine the data on the California stores. Notice that we're comparing two products across five stores, in an easy-to-understand format. We did this by navigating the data presented by Jaspersoft OLAP without having to design and run a report.

Dimensions					Product	
					All Products	Drink
Measure	Store	Store Country	Store State	Store City	Beer	Wine
Store Sales	All Stores	USA	CA	Alameda		
				Beverly Hills	34.44	127.60
				Los Angeles	44.88	140.52
				San Diego	43.82	91.06
				San Francisco		12.78

Filter: Month=[Time].[2006].[Q4].[12]

Figure 5-8 Pivoting the View

```
select [Product].[Drink].[Alcoholic Beverages].[Beer and Wine].Children ON COLUMNS,
Crossjoin({[Measures].[Store Sales]}, [Store].[USA].[CA].Children) ON ROWS
from [Sales]
where [Time].[2006].[Q4].[12]
```

We now see that the San Francisco store sold a fraction of the alcoholic beverages that other California stores sold, and that it sold no beer at all.

5.1.3.2 Other Approaches

In the course of this example, we have unearthed questions we hadn't thought to ask previously: is the lack of beer sales part of a trend in this store's overall beverage sales? We can answer other questions, such as whether the store's overall sales lag across all products, or whether another product's sales make up for the lack of beer sales.

Alternately, we could look at the best performing stores in each category or product to try to understand the secrets to their success. Ranking and sorting can be very helpful in this type of analysis when comparing many values. Your MDX statements can include any imaginable statistical functions. For example, you could include arbitrarily-complex expressions to show the top or bottom percent of any measure.

```
select [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and
Wine].Children ON COLUMNS,
Crossjoin([Store].[All Stores].[USA].[CA].Children, {[Measures].[Store Sales]}) ON
ROWS
from [Sales]
where [Time].[2006].[Q4].[12]
```

We can show analysis data as a graphical chart, just as might be defined in a report. This chart shows the same data as the sample above, but in a vertical 3D bar chart

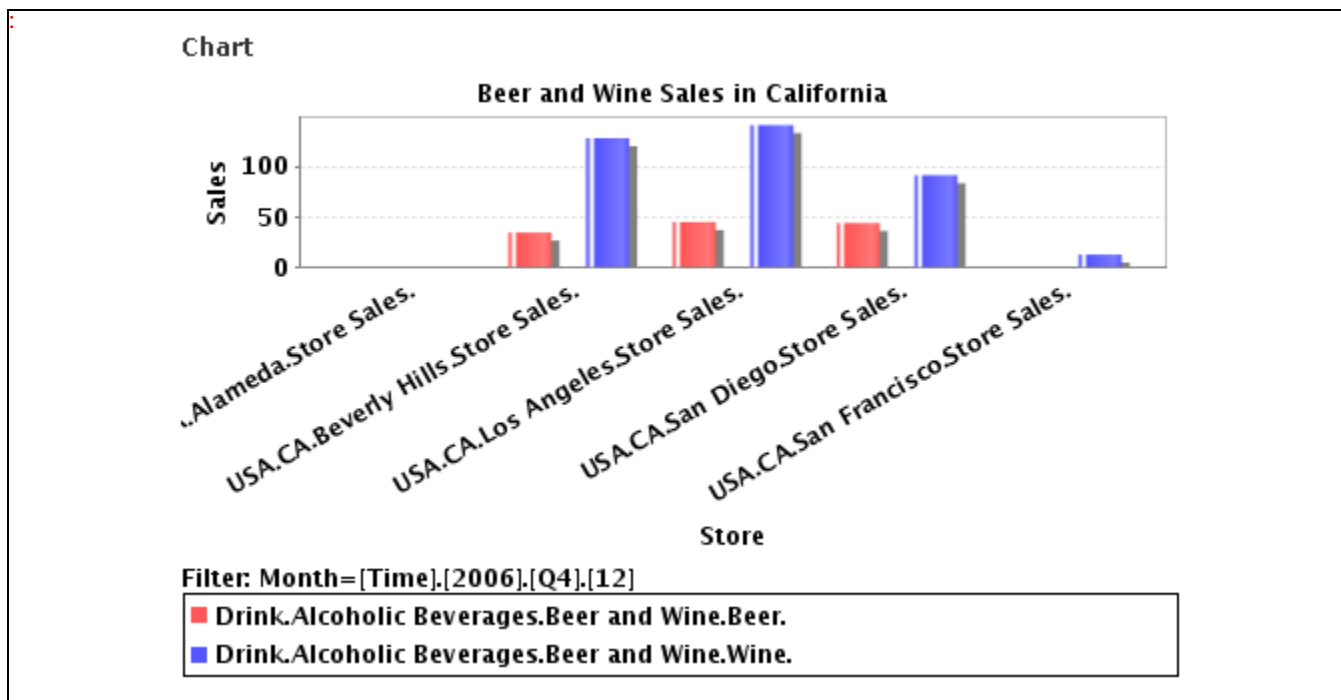




Figure 5-9 Chart of Beer and Wine Sales in California

The chart focuses on aggregated data, that is, the totals for all the sales at any dimensional coordinates. This is not to imply that analysis supplants transactional reporting. Analysis provides transactional reporting on demand by showing your data in different contexts and different levels of roll-up.

Examining the basic row data is still very helpful in furthering your understanding of a particular area. For example, in the

table, use  to drill-through to the wine sales for San Francisco to show detailed transactional information. We're looking at individual rows of data to understand this one store's wine sales in December of 2006 (chart is truncated).

Foodmart Sample Analysis View

 **Drill Through Table for [Store_Sales = 12.78]**

Store_State	Store_City	Store_Name	Store_Sqft	Store_Type	Year	Quarter	Month	Week	Day	Product_Family
CA	San Francisco	Store 14	22478	Small Grocery	2006	Q4	12	1	18	Drink
CA	San Francisco	Store 14	22478	Small Grocery	2006	Q4	12	1	18	Drink
CA	San Francisco	Store 14	22478	Small Grocery	2006	Q4	12	2	24	Drink
CA	San Francisco	Store 14	22478	Small Grocery	2006	Q4	12	50	4	Drink

Figure 5-10 Drill-through Table of San Francisco Beer And Wine Sales

5.2 Maintaining the System

This section includes information about system maintenance, including:

- [XML/A Definitions](#)
- [Monitoring the System](#)
- [Maintaining Tables](#)
- [Clearing the Cache](#)

5.2.1 XML/A Definitions

XML for Analysis, or XML/A, uses SOAP (Simple Object Access Protocol) to define XML message interfaces for the client application to interact with the database server for online analytical processing, or OLAP, over the Internet. Based on MDX, XML/A provides a standard language interface for sending and processing OLAP requests. For more information, see <http://www.xmla.org/nonmemdefault.asp> or <http://msdn2.microsoft.com/en-us/library/ms187178.aspx>.

The advantage of XML/A is that it is agnostic regarding query languages and data sources. That is, XML/A provides a uniform interface to various multidimensional servers, such as Microsoft OLAP server and Mondrian OLAP server, and to various data sources, such as MySQL or Microsoft SQL Server.

JasperSoft OLAP provides XML/A support in terms of XML/A definitions for discovering multidimensional metadata, XML/A connections for accessing multidimensional data sources using SOAP, and XML/A OLAP views for querying multidimensional data.

One popular use of XML/A is to allow Excel users to slice and dice OLAP data through Pivot Tables with an ODBO driver. JasperSoft ODBO Connect is such an ODBO driver, which can connect to JasperSoft OLAP, Mondrian and Microsoft Analysis Services. You can purchase ODBO Connect from JasperSoft's store at <http://www.jaspersoft.com/jaspersoft-odbo-connect>

5.2.1.1 XML/A Sources

The XML/A source defines an XML/A catalog, or schema, with an OLAP connection for discovering OLAP metadata, such as cubes, dimension, and measures.

To create an XML/A definition:

1. On the Home page, click **View > Repository** to open the repository.
2. In the Folders pane, right-click the /Analysis Components/xmla/definitions folder.
3. In the menu that appears, click **Add Resource > Mondrian XML/A Source**.
The Set Mondrian Source Properties page appears.
4. Enter the following information (you can change the supplied information if necessary):
Name: Test Mondrian XML/A Source
Resource ID: [supplied automatically]
Description: Test of adding Mondrian XML/A source
Catalog: Foodmart [or another name]
Mondrian Connection Reference: [supplied automatically]

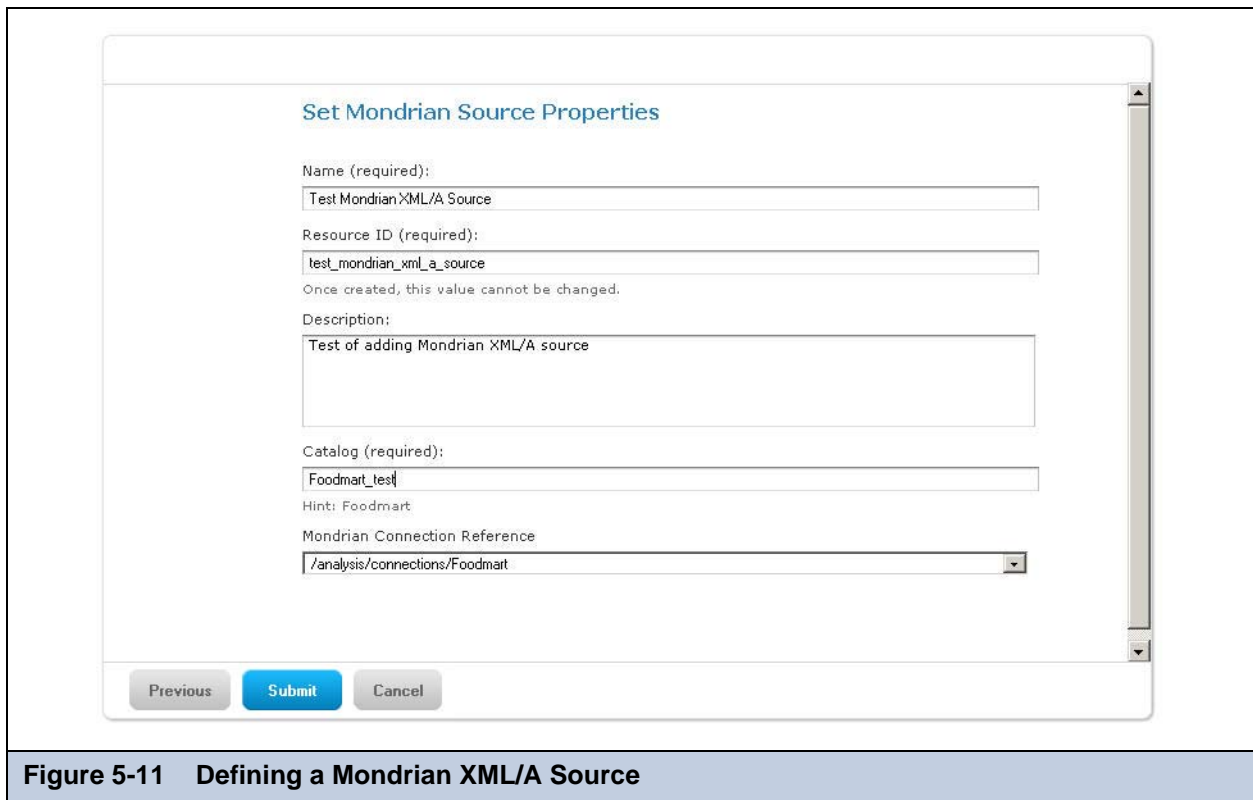


Figure 5-11 Defining a Mondrian XML/A Source

5. Click **Submit**.
6. The new definition appears in the repository.

5.2.1.2 XML/A Connections

An XML/A connection is an OLAP connection for accessing a multidimensional data source using the XML/A interface. It contains specifications for catalog, multidimensional server, data source, HTTP server URI, user name and password, as described in 2.3.5, “Creating a Mondrian Connection,” on page 20.

Specifications in XML/A Connections	
Specification	Explanation
Catalog	XML/A catalog found in the XML/A definition containing the multidimensional metadata, such as cubes, dimensions, measures, levels.
Data Source	Multidimensional data source defined by its provider, such as Mondrian, and database name, such as Foodmart
URI	HTTP URI to JasperReports Server, suffixed with \xmla
User Name	JasperReports Server login
Password	JasperReports Server password

To create an XML/A connection:

1. On the Home page, click **View > Repository** to open the repository.
2. In the Folders pane, right-click the /Analysis Components/Analysis Connections folder.
3. In the menu that appears, click **Add Resource > OLAP Client Connection**.
The Set Connection Type and Properties page appears.
4. In the Type drop-down, select **XML/A** and enter the following information in the fields (you can change the supplied information if necessary):

Name: Test Foodmart XML/A Connection
Resource ID: [supplied automatically]
Description: Test of adding XML/A connection
Catalog: Foodmart [or another name]
Data Source: Provider=Mondrian; DataSource=Foodmart
URI: http://localhost:8080/jaspersoft-pro/xmla
User Name: jasperadmin [or your username]
Password: jasperadmin [or your password]



You can leave the **User Name** and **Password** fields blank so the logged in user's credentials are passed to the remote server when the connection is accessed.

Add OLAP Client Connection

First select the type of connection you want to add, then enter the required property values.

Type: XML/A

Name (required): Test Foodmart XML/A Connection

Resource ID (required): test_foodmart_xml_a_connection

Description: Test of adding XML/A connection

Catalog (required): Foodmart

Hint: Foodmart

Data Source (required): Provider=Mondrian; DataSource=Foodmart

Hint: Provider=Mondrian; DataSource=Foodmart;

URI (required): http://localhost:8080/jaspersoft-pro/xmla

Hint: http://localhost:8080/jasperserver-pro/xmla

User Name: jasperadmin

Password:

Previous Submit Cancel

Figure 5-12 Defining an XML/A Connection

5. Click **Submit**.

The new definition appears in the repository.

5.2.1.3 XML/A and OLAP Views

You can create an OLAP view against an XML/A connection as you did against a Mondrian connection (see [3.1, “OLAP Views,” on page 27](#)). The steps are nearly identical. Simply select an XML/A connection instead of a Mondrian connection when prompted for the connection.

5.2.2 Monitoring the System

Jaspersoft OLAP uses standards-based technology, so the maintenance of the application itself is similar to that of any web application. The important considerations are reliability (making sure the application servers and DBMS are up and running) and performance.

Jaspersoft recommends standard monitoring tools for production deployments that have rigorous availability requirements. Besides doing a basic check of system health, occasional monitoring of the amount of memory being used by the OLAP engine can be useful for optimizing performance depending on your usage. If the OLAP engine is consistently using close to the maximum amount of heap memory allocated, it could be caching more if more memory were allocated to that JVM process.

When you notice problems, your first step is generally to search the logs for clues. JasperReports Server uses standard log4j for all system logging, and all output is directed to WEB-INF/jasperserver.log by default.

- If there are FATAL messages in the log, the cause is usually related to the configuration of the web application and is something that keeps the application server from loading the web application.
- If there are any ERROR messages in the log, they denote an unexpected error within the application and should be investigated.
- Messages at the WARN level do not indicate a serious problem, but they may contain useful information that something may be wrong with the setup, that could lead to errors.
- DEBUG and INFO messages may help developers or expert users of the system, but may be safely suppressed by setting the log4j.rootLogger to WARN messages and above in the log4j.properties file. When debugging an exception, set the logging level to DEBUG for the specific classes mentioned in the exception.
- The log4j documentation is available at: <http://logging.apache.org/log4j/docs/manual.html>.



For information on contacting Technical Support, see 5.5, “**Troubleshooting Information for Technical Support,**” on page 98.

5.2.3 Maintaining Tables

In an OLAP schema, physical tables are often designed for performance, and there are some tables whose data is derivative of other tables. It is important to keep the derived tables up-to-date. For example, you can build aggregate tables from the fact tables that they summarize; they would contain the same data aggregated to a higher level in some dimension hierarchies. Then join the two fact table in the database; this solution performs better than virtual cubes. Closure tables list out all the relations among the levels in underlying parent-child tables, which improves performance.

There are several approaches to maintaining derived tables:

1. Materialized views can be a useful tool for this task in databases that support them, such as Oracle. The SQL for building the tables simply needs to be expressed as a materialized view, and they are rebuilt as needed from the underlying tables.
2. When the same data-loading process occurs on a daily cycle, having a scheduled routine rebuild the derived tables may help. The simplest implementation of this approach is to have a SQL script that drops and rebuilds these tables, which runs at a certain time of day. If the tables are very large, refine the process by incrementally rebuilding only the changes’ this potentially requires more complex queries.
3. Triggers may be appropriate if the timing of new data is unpredictable. One variation of this approach would be similar to the approach in the previous step, except that the data-loading job would update a status table at the end, which would fire triggers to rebuild all the changes to derived tables. Another variation would be to create one trigger per derived table, which updates/rebuilds it when the underlying tables change. This latter variant might not be appropriate when the data changes frequently, because of the amount of time that would be spent continually rebuilding these tables.

One of these approaches might be better suited to your data warehouse.

The data-loading job itself can rebuild the derived tables after loading new data. If you are using an ETL tool, this may be a convenient way to unify the tasks of table maintenance and data-loading. The specifics of this depend on the tool used.



It is often better to disable constraints (indexes) before rebuilding a table’s data and to re-enable the constraints afterwards than to rebuild the table with the indexes enabled.

After rebuilding or significantly changing the data in any tables, make sure that the statistics for the database optimizer are up-to-date, which may involve gathering or estimating statistics on tables or the entire schema. This applies only to databases with statistics-based optimizers, such as Oracle or DB2.

Periodically archiving data that is no longer actively used is an important part of a long-term maintenance plan. For example, if a company only regularly reports on the last year's data, except for quarterly historical reports, data that is older than one year can be moved from the fact tables into archive tables. Quarterly aggregates can be built from these archive tables. These smaller fact tables and aggregate tables will perform better.

Another table maintenance task is taking regular backups to protect the data. There are various standard approaches to this which may depend on your database. Fact tables may be very large, but the amount of new data each day may be relatively small in comparison, and often only additive, suggesting that an incremental backup strategy can be well suited for OLAP.

5.2.4 Clearing the Cache

Jaspersoft OLAP caches OLAP query results and metadata (such as cube structures and data source connection information) in memory. Cached items are removed automatically through JVM garbage collection, where the least-recently-used items are cleared first. However, when new data is loaded into a database accessed by Mondrian connections, the cache must be cleared; otherwise, OLAP query results cannot include the new data. The cache must also be flushed when you modify Mondrian client definitions or their data sources.

You can clear the OLAP cache in any of these ways:

- Login with `superuser` privileges and display the OLAP Settings page.
In the OLAP Settings panel, click **Flush OLAP Cache**.
- Using utilities such as `cURL` and `wget`, send an HTTP request to this URL:

```
http://<server>:<port>/<context-path>/olap/flush.html?j_username=superuser
&j_password=<superuser_password>
```

- Restart the application server instance.
- Call the Java API.

For more information about using HTTP requests, refer to [Chapter 6, “Executing OLAP Views Using the HTTP Interface,” on page 99](#), and the *JasperReports Server Ultimate Guide*. For a comparison of caching in JasperReports Server and Jaspersoft OLAP, see the *JasperReports Server Administrator Guide*.

5.2.4.1 About Multiple Server Cache Coordination

When multiple OLAP engine servers share a single repository, you can coordinate the flushing of their caches. In this case, changes in one server trigger notifications to the other servers that are connected to the same repository. They in turn flush the affected Mondrian caches. Internally, Jaspersoft OLAP uses Ehcache. You can configure it by editing the `WEB-INF/olap-ehcache.xml` file. See http://ehcache.org/documentation/distributed_caching.html for details.

There are a variety of options for working with a distributed cache, including:

- RMI (Remote Method Invocation).
- JGroups
- Terracotta
- JMS (Java Message Service)

The Ehcache server is the simplest option.

By using the JasperReports Server Java API, a process external to the application servers can trigger coordinated cache flushing. To do so:

- Ensure that the external process and the application servers share the same repository and cache configuration.
- Use the `OlapManagementService.notifySchemaChange(ExecutionContext context, Resource resource)` method. The `context` can be null, and the `Resource` (which must be in the repository) can be a Mondrian connection or a data source used by Mondrian connections. If the resource is data source, all Mondrian connections that use that data source are flushed.

For further details on the Java API for flushing the cache, refer to the *JasperReports Server Ultimate Guide*.



After clearing the cache, it is a good idea to preload the empty cache by issuing MDX queries with a script or web automation tool. This populates the cache before your users begin analyzing data and can improve performance. This is sometimes known as *pre-caching*.

For information about the JasperReports Server API, refer to *JasperReports Server Ultimate Guide*.

5.2.4.2 About Load-balanced Environments

In a load-balanced deployment, you can clear the caches of each OLAP engine server in turn, so that the overall environment has no downtime.

To clear each cache in turn:

1. For each server in the load-balanced environment:
 - Make the server inactive in the load-balancer environment.
 - Clear the server’s cache.
 - Refresh the cache by running MDX queries so that your users enjoy fast response times. This can be done by sending XML/A requests to the server.
 - Add the server back to the active load-balancer set.

For more information on load balancing, including a diagram, see [5.3.3.5, “Load Balancing,” on page 95](#).

5.3 Tuning Performance

5.3.1 Understand Jaspersoft OLAP Performance

This section provides a framework for performance tuning of Jaspersoft OLAP for the technically skilled reader who understands the basics of SQL and RDBMS tuning and has used the Jaspersoft OLAP interface to perform some queries.

The first step is to gain a high-level understanding of the common performance bottlenecks in a typical installation of Jaspersoft OLAP. The user submits an MDX query, either through the MDX query editor or implicitly by taking an action in the navigation table, such as drilling-down, expanding, and doing cross-joins. This MDX query is passed to the OLAP engine, with either a local Mondrian connection, or over a network using XML/A. The OLAP engine then processes the query, retrieving whatever it can from its memory cache and generating SQL queries to look up anything else from the database. These SQL queries are then executed on the database server, and the results are returned to the OLAP engine and may be cached in its memory space. The OLAP engine further processes the data to assemble the SQL and cache lookups into an OLAP Result object. The result is then returned to the front-end by the same means that it received the request: either locally in-memory or using XML/A. The front-end server does further processing to layout the new view.

To summarize, the following table described the complete round trip MDX request and response in Jaspersoft OLAP, along with commentary on which computing resources are most relevant for bottleneck analysis:

Steps in an MDX Query & Response		
Step		Resources Relevant to Bottleneck Analysis
1	MDX is generated	CPU and memory of front-end negligible
2	MDX is passed to local OLAP engine	
	MDX is passed remotely using XML/A	Network latency between front-end and OLAP engine
3	MDX is processed by OLAP engine	CPU and memory of OLAP engine
4	SQL sent to database server	Network latency between OLAP engine and database
5	SQL queries executed by database	Disk I/O, CPU, memory of database server

Steps in an MDX Query & Response, continued		
Step		Resources Relevant to Bottleneck Analysis
6	SQL results returned to OLAP engine	Network latency and bandwidth between OLAP and database
	OLAP Result object constructed	CPU and memory of OLAP engine
7	Result passed to local front-end	Memory of engine/front-end
	Result returned using XML/A	Network latency and bandwidth between OLAP and UI
8	Result is processed and displayed	CPU and memory of front-end

One question about performance that may come up here regards the difference between local Mondrian connections and XML/A. With Mondrian connections, the front-end and OLAP engine routines are running in the same application server, sharing the same memory and CPU resources. While this happens serially for each single request, the requests of multiple users occur in parallel. So, in multi-user installations, there is a performance gain to be made by separating the front-end server from the OLAP engine server using XML/A. XML/A incurs the additional processing overhead of converting requests and responses to XML, and then transmitting them through HTTP, but this is quickly outweighed by the performance benefits of parallelism when there are multiple simultaneous users. Another advantage of XML/A in performance and scalability optimization is that a front-end can have its request load balanced using HTTP to one of many OLAP engines in a cluster. [5.3.3.5, “Load Balancing,” on page 95](#).

5.3.2 Profiling Performance



This section describes functionality that can be restricted by the software license for JasperReports Server. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

A basic observation about performance optimization is that improving the speed of any subsystem speeds up the overall system only in proportion to the amount of time that is spent in that subsystem as compared to the overall duration. This is sometimes referred to as Amdahl's Law. The law suggests an approach for iterative performance tuning, where the first step is to observe how much time is spent in each subsystem and therefore where the biggest improvements to the overall system can be made. After improvements are made to a subsystem, the whole system should be profiled again and the process repeated.

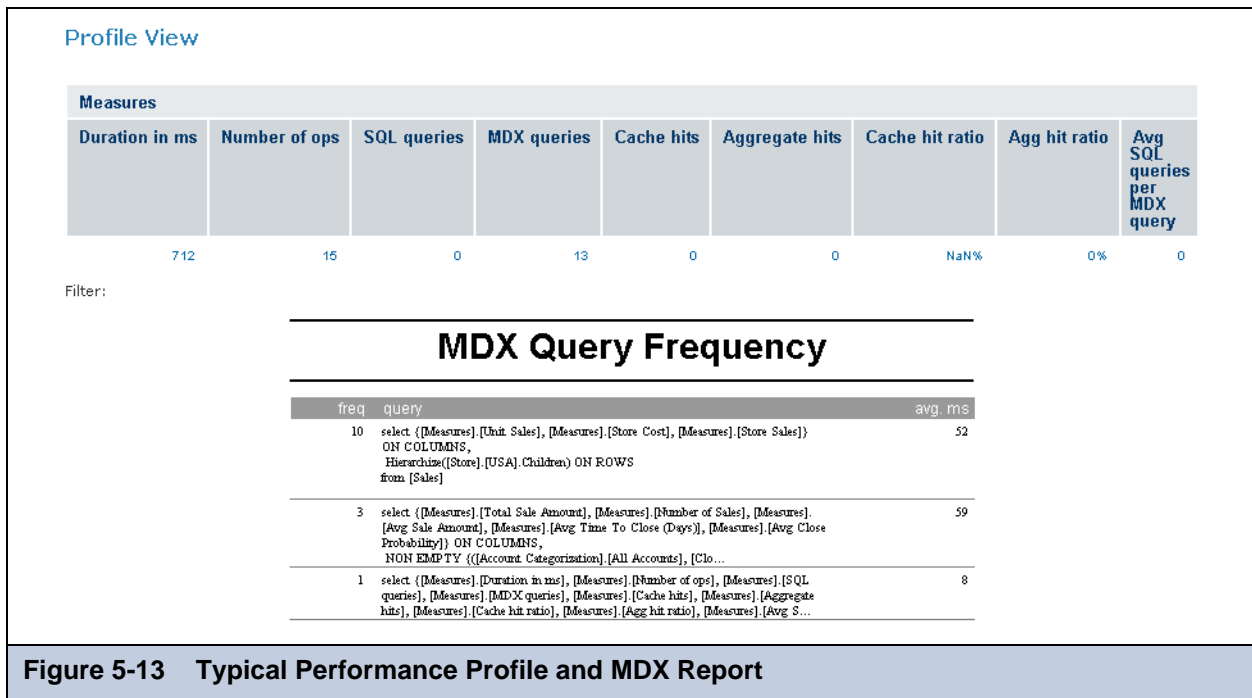
In ROLAP implementations such as Jaspersoft OLAP, one quickly finds that most of the system time is spent running SQL queries. The most effective ways to improve this time are (a) tuning the database for the common queries and (b) increasing the size of the OLAP engines' available memory for caching, which reduces the number of queries that must be sent to the database itself. JasperReports Server commercial editions have a profiling feature that can be used to save performance statistics for any part of the system code. This profiling can be used to analyze MDX and SQL execution times.

When the profiling feature is enabled, every query is timed and the timing data is stored in the ProfilingRecord table and displayed in performance reports and views. The reports are in <organization>/Performance/Reports. The views are in <organization>/Performance/Views.

To use the profiling feature:

1. Login with `superuser` privileges and display the OLAP Settings page.
2. In the OLAP Settings panel, select **Performance Profiling Enabled** and click **Change**.
3. Review the results to determine if all the data that you need was recorded.
4. Change the settings in the OLAP Settings panel as needed and analyze the results again.

Sample profiling OLAP schemas are included with the product in /Analysis Components/Analysis Schemas, as are some sample reports and views showing MDX and SQL timings (/Performance). The samples can be modified and used to create your performance reports, or they can serve as templates for entirely new reports. [Figure 5-13](#) shows two examples of their output.



Enabling performance profiling may have a minor impact on the overall performance of the system, since Jaspersoft OLAP must run additional queries in order to time each query. Therefore, we recommend disabling performance profiling in production environments, unless you are actively performance tuning. The profiler is designed to minimally impact the queries it times; the overhead of generating and saving objects to the database occurs outside of the execution timing of the target queries.

There are several approaches performance profiling, depending on your goal:

- To get a general sense of the mix of queries being performed and their average response time, collect data from actual users’ activities. This is useful for observations about what the most frequent queries are and what the slowest queries are.
- To investigate the cause of individual queries that are slow, repeat the same query in various conditions, such as load on the application servers or database server.
- To develop better MDX, try rewriting an MDX query and looking at the generated SQL timings for the various approaches.

Having a test automation tool (such as Selenium) simulate users performing queries will be very helpful, so that you can compare results for the same actions with different settings for the database, application configuration, and so on.

After viewing and analyzing the query data, you can make some controlled changes and collect timing data again, holding other variables constant, to determine if your changes result in an improvement for the sample set of queries. Having a large sample set is desirable so that you don’t overly tune for a particular usage at the expense of others. A standard set of data and sample queries may be referred to as a “benchmark,” and having a benchmark is an important part of effective performance tuning projects.

5.3.3 Jaspersoft OLAP Tuning Process and Options

Building a high-performance database for the largest scale of data volume can involve various optimizations which are categorized here as the physical hardware level, the Relational Database Management System (RDBMS) level, the OLAP engine level, and the logical schema level. Each of these levels has its own complexities and requires domain knowledge to make the proper decisions, so it is not uncommon that a large-scale data warehouse project would involve experts in each of these areas.

5.3.3.1 Hardware Level

The following comments are meant to provide general advice on optimizing hardware in an RDBMS:

Optimizing Hardware in an RDBMS	
Optimization Issue	Comments
Disk I/O, bus speed	Hardware used for the database server should be selected for latency and bandwidth of data transfer. RAID striping is an example of a hardware optimization that should be considered, unless specifically counter indicated by the RDBMS vendor, as for example DB2 does. RAID mirroring can have a negative impact on I/O performance for some database systems, so a backup solution is preferable to redundant disks from a performance tuning standpoint.
File systems	For the largest, most scalable implementations, some vendors offer specialized file systems for supporting RDBMS systems.
Clustering	Some database vendors have support for clustering, which allows for a single database instance to use multiple physical servers, both for performance and uptime reasons.
CPU and memory	CPU usage of the database server typically comes into play to a greater extent for OLAP than for OLTP database applications, both because of the number of large joins usually involved in multidimensional queries and because of the number crunching functions like summation that are often applied to calculate aggregated measures. Memory can be used by the database server both for computing joins, and for caching SQL results and indexes.

5.3.3.2 RDBMS Level

The biggest decision at this level is which RDBMS to use. Oracle, MySQL, Postgres, MSSQL, DB2 and so on all have their own trade-offs and offer different performance tuning options. Comparing them is beyond the scope of this document. Each of these vendors provides its own information on performance tuning and recommended operating system settings. The following comments are meant to provide general advice on optimizing an OLAP project in an RDBMS:

Optimizing an OLAP Project in an RDBMS	
Optimization Issue	Comments
Temp space	The number of large joins required by multidimensional queries makes heavy use of temp space for storing intermediate results. A SQLException is thrown when the amount of available temp space is exceeded, and the space should be increased sufficiently to avoid this. This should be tested with a realistic number of simultaneous users.
Data spaces and index spaces	Having separate devices for the data space and index space is important in synchronizing the use of indexes for faster lookups, which becomes especially important for fact tables. Having a separate device for database log files is also recommended.
Memory block sizes	OLAP applications can often benefit from bigger block sizes than for OLTP applications, primarily for loading the data, but also for retrieval.
Caches	Experiment with changing the sizes of various database caches and buffers, and using profiling tools to analyze the results.
Table partitioning	Some RDBMS, for example Oracle Enterprise Server, support partitioning, so that one logical fact table with several million rows can be separated into different physical tables.

Optimizing an OLAP Project in an RDBMS, continued	
Optimization Issue	Comments
Statistics	Some RDBMS incorporate various types of optimizers. For example, two common kinds are statistics-based optimizers or cost-based optimizers. For Jaspersoft OLAP, we recommend statistics-based optimizers, and statistics should be generated or estimated periodically, or when there are significant changes to the indexes or data.
Materialized views	Oracle includes a feature known as materialized views, which are queried and automatically updated as normal views, but are backed by actual physical tables, meaning they don't have the added cost of recomputing the query that makes up the view. These are especially useful for building aggregate tables in an Oracle installation, so that they are automatically updated when new data is loaded to the underlying fact and dimension tables. Otherwise, aggregate tables must be rebuilt, re-indexed and re-analyzed after each new data load.
Read-only/undo logs	Using read-only connections for OLAP may allow for faster cursor access. Disabling undo and redo logs for the OLAP schema may also be appropriate, if the data is simply a copy from the original data source.

5.3.3.3 Schema Design and Optimization Level

Whatever the hardware and RDBMS choices, the logical schema design has a large role in determining the performance of a data warehouse, and often a bad logical design cannot be improved significantly by any amount of hardware and RDBMS tuning. Furthermore, optimizations at the schema level, such as indexes, give the RDBMS the greatest amount of information to build the proper query plans for efficient queries.

Keep the following in mind when designing a new schema for Jaspersoft OLAP:

- Snowflake schemas (where one or more dimension tables have a “subdimension” table) are not as efficient in returning queries as non-snowflake schemas.
- For best performance in a star schema, it is better to construct each dimension as a single table, even if this means having redundant information in a column. For example, location (city, state, country) is better than city_loc (city, state) and state_loc (state, country).
- Virtual cubes (where multiple fact tables are joined together as if a single cube) pay a performance penalty on each query, to do this join. For best end-user performance, it is preferable to build the joined fact table during data-loading time, or instantiate it as a materialized view in Oracle.
- Build closure tables for parent-child relationships. For example, use a closure table for an employee_id and a supervisor_id that references the supervisor’s employee_id. Without a closure table, multiple SQL queries would be required to lookup an ancestor or descendent more than one level away. By building a closure table, which has a row for every combination of employee_id and ancestor, along with the distance of the relationship, it is possible to lookup a relationship of any distance with a single SQL query, which is how Jaspersoft OLAP will generate the query for the corresponding MDX.
- One of the most important variables in determining the duration of a multidimensional query is the size of the fact table involved. Depending on the nature of the query, the duration of the query may be more than directly proportional (polynomial) to the size of the fact table. In this case, it would be quicker to query two fact tables half the size, then take the union of the results.

Also, if a large fact table has many rows that are not commonly queried, it makes sense to split them into another table, making a logical partition. For example, if most queries about product orders are only concerned with the current year, putting order information for other years into separate fact tables would be an effective performance optimization. The schema might have an orders_2007 and orders_historical table, and the orders_historical table could be further split into tables for each historical year.

- Use NOT NULL constraints wherever applicable. Specifying this for a column not only helps as a check on data integrity, but allows the RDBMS to speed up joins involving these columns.
- Properly indexing columns can make the single biggest difference in optimizing multidimensional queries. All foreign keys should be indexed, with particular attention to the dimension IDs in the fact tables. RDBMS like Oracle allow for compound indexes, which are recommended for OLAP. The order of the columns in a compound index matters, so there should be compound indexes for each ordering that corresponds to the order of joins in queries.

- Aggregate tables can be used to build a higher-level fact table, where the data is rolled up and there are fewer overall rows. For example, a fact table that has one row per second could have an aggregate table with one row per day. Maintain your aggregate tables using your data load processes (such as JasperETL jobs). Usually, after loading a fact table, the data load process empties the aggregate tables and repopulates them with the results of aggregate queries that are run against the refreshed fact and dimension tables.

Determining the best set of aggregate tables to use involves weighing a series of trade-offs. Having many aggregate tables is often a waste of time and disk space, as what aggregate tables can be used varies according to what MDX is being run. There is a desktop tool from the Mondrian open source community, the Aggregation Designer, which uses statistical analysis of the underlying database combined with the schema to determine what are the best aggregate tables to have. The Aggregation Designer can update your schema XML file with new aggregate definitions which can be used by Jaspersoft OLAP and also generates SQL to create and populate the suggested aggregate tables, which can be used as part of your data load processes. You can download the Aggregation Designer at <http://jasperforge.org/projects/jasperserver/downloads>.

For more information, refer to the *Mondrian Technical Guide*.

- In the schema XML file, a Level tag may have an `approxRowCount` attribute. If you can provide a reliable estimate of the number of rows at a given level, `approxRowCount` reduces the time required to determine the size of the level at query time.
- Different MDX queries can achieve the same results but with different performance characteristics. There are no simple rules about which queries perform better. Instead, you will have to use trial-and-error approach to determine the best ones for your system.

5.3.3.4 OLAP Engine

There are a number of system properties which control the behavior of the OLAP engine. These properties control decisions about the algorithms used and limits on resources. They may be modified on the Analysis Options page in Jaspersoft OLAP Community Edition, as part of an iterative tuning process. For more information on these properties, see the *Jaspersoft OLAP User Guide*.

5.3.3.5 Load Balancing

For systems with many simultaneous users, multiple application servers and load balancing will keep response times close to those for a single user with a single application server. Both the front-end and OLAP engines can be distributed in a load-balancing setup. The OLAP engine does more of the number crunching and “heavy lifting” than the front-end, so it is especially important to have multiple engines when there are many simultaneous users of a deployment. To ensure that a single failure cannot limit access to the data, there should be at least two front-end application-servers on different computers and two OLAP engines on different computers. The complete Jaspersoft OLAP software will be installed similarly on each computer, but the metadata will be configured slightly differently.

For example, consider the scenario where there are to be two front-ends and two back-ends. If there are sufficient resources, you can put each front-end or engine on a separate computer. If you only have two computers, you can put one front-end on each of the two, and one back-end on each of the two. Users will be directed to either of the front-ends through some URL with a standard external load-balancing tool such as an apache web server with `mod_jk` or an internally redundant network appliance. Both of the front-ends will be load-balanced so that their requests are routed to either of the two engines. The requests from the front-ends to the back-ends will be transmitted through HTTP using the SOAP protocol to deliver XML/A requests (containing MDX queries) and responses (result sets).

Let’s say the front-end application servers are running on `server1:8080` and `server2:8080`, and the back-end application servers are running on `server1:9080` and `server2:9080`. In a typical setup, all four of these application servers should use the same JasperReports Server repository for metadata, and same data sources. You should configure the front-end load-balancer to direct traffic between `server1:8080` and `server2:8080`. You should create an XML/A Connection using the Jaspersoft OLAP interface with a virtual host as the server in the URL name, for example `http://xmla.yourdomain.com`. This URL should be directed to load balance between `server1:9080/jasperserver-pro/xmla` and `server2:9080/jasperserver-pro/xmla`. This effectively splits the load from simultaneous users across the two computers evenly.

In addition to the performance and scalability benefits of load balancing, there are also uptime benefits. The above network architecture removes any single point of failure, other than the database. Ensuring redundancy in the database server can be achieved with clustering and hot, warm or cold backups, depending on the RDBMS. Consider an operation which must be up 24 hours 7 days a week. When the new data is loaded overnight, the OLAP engines cache must be cleared and reloaded. If

there are multiple OLAP engines in a load-balanced deployment, each of these can be taken offline and refreshed by itself. This same technique of taking an OLAP engine or front-end out of the load-balancing mix of active servers can be used for other types of scheduled maintenance, and to reduce the end-user impact of unexpected outages.

Figure 5-14 depicts a load-balanced OLAP environment.

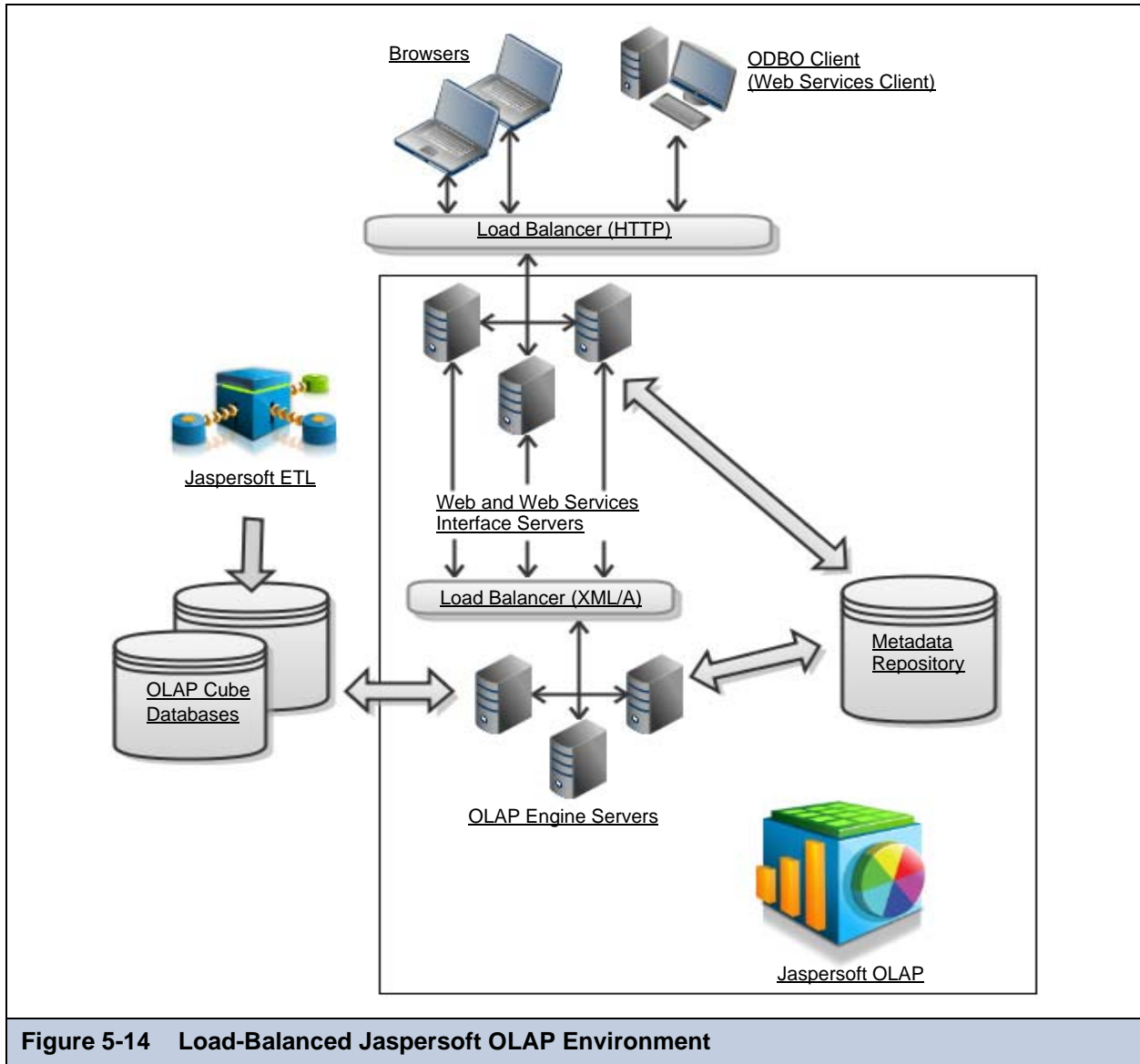


Figure 5-14 Load-Balanced Jaspersoft OLAP Environment

5.4 Integrating Jaspersoft OLAP in the Enterprise's Data Flow

The BI data warehouse is only one part of an overall store of data in an enterprise. Data is generated and collected in one or more data sources. Often, the interesting and useful bits of information are extracted, transformed and loaded into a centralized repository. In a large company, user information is typically centralized as well, and various applications use some sort of authentication service to provide a unified login process for all applications, rather than duplicating user names and passwords across them. Server applications, like JasperReports Server, usually have their own persistent metadata repository that records choices with settings, preferences, and configurations of the application. The focus for this section is on how the data used for analytic reporting fits into this complex enterprise data flow.

What are the user's expectations about new data? The users' requirements related to data flow integration generally have to do with availability and consistency of the data. Availability, for purposes of this discussion, has to do with how long it takes to see new data in the reports. Consistency refers to whether or not the reports reflect a valid set of data as a snapshot of some

time. A further suggested requirement is that the users' view of the data through reports is consistent in a chronological progression, so that older snapshots are never shown after more recent snapshots. Often there are trade-offs between these requirements, for example, to guarantee consistency, it may be necessary to wait on some data before making it available.

When trying to ensure that the data flow supports consistent reporting, a few aspects of the OLAP schema should be considered:

- Facts reference dimensional members. This means that rows in a fact table refer to rows in dimensional tables. With proper referential integrity, there should be a foreign key constraint on each column of a fact table that points to an ID column of a dimension table. This implies that for data consistency, updates to the dimension tables should be completed before loading new data into the fact tables that reference them.
- Some calculated measures may depend on data from multiple different sources. For example, let's say there is a measure representing sales per hour by employee. The number of sales comes from the transaction data source, and is loaded into the data mart every night. The number of hours worked for each employee comes from an HR system, and for the sake of this example, let's say is imported once per payroll cycle. This means that a naive implementation of the sales per hour measure will be inconsistent, because the sales are reflected up to yesterday, but the number of hours worked may be weeks out of date. If the data cannot be loaded with the same frequency to be consistent over all time periods, the calculated measure should be changed to take this into account and not show inconsistent results for the current period. Publishing the times when the data is refreshed from various sources will be useful so that analysts can properly construct such calculated measures as in this example.
- Even with a single data source, if the data is being queried as new data is coming in, the results in a report may mislead the viewer into thinking that all of the data has been loaded. The easiest solution is to suspend reporting while the new data is coming in. If this is not feasible, it may be possible to import the new data in a single database transaction, so that it will not be visible until the commit has completed for the entire batch.
- Aggregate tables, closure tables, joined cubes, and other derived tables should be updated or rebuilt after loading all the data in the tables they depend on, and before they are used for live reporting. Please see the administration and maintenance section of this guide for strategies on how to maintain those tables.

Remember that the user's view of the data is also affected by the memory cache in the OLAP server. The cache holds some, but not all, of the data since it was first queried. Loading new data while a server has already cached some of the data from the previous state of the data warehouse leads to an inconsistent state. These approaches to dealing with caching effects on data inconsistency are generally helpful:

- Disable the memory cache for the OLAP servers while loading new data that would affect the results
- To do so, log into the web application as `superuser` and display the OLAP Settings page. In the OLAP Settings panel, select **Disable OLAP Memory Caching**. This takes the cache out of the picture, so as long as the underlying data is in a consistent state as it is loaded in, the reporting will be consistent with the data. For large amounts of data or numbers of users, this is not recommended, because the performance of viewing un-cached reports with every request will be slow. This option is best suited for developing or testing data loading, rather than for production usage.
- Disable access to reporting during data loading. This approach is quite simple, and well-suited if the data-loading occurs when the system is not being used by data analysts, for example, overnight. Remember to clear the cache (see [5.2.4, "Clearing the Cache," on page 89](#)) after the new data has been loaded and pre-cache with some queries against the new data.

We have seen a number of ways in which a report may show inconsistent information when it reflects only a partial snapshot of a data load. The simple solution of generating and viewing reports only when the data is not being loaded is sufficient for most cases. But, sometimes there are business requirements to be able to view reports and perform analysis tasks while data is being loaded. In these cases, satisfying this requirement while preserving the data consistency requirements involves a more complex solution. The technique is sometimes called "trickle and flip," a phrase coined by industry pioneer, Ralph Kimball.

The basic idea in trickle and flip is to have duplicate fact tables during loading, with the reporting applications pointed to the first version of the table, while new data is being loaded into the second version of the table. Once some or all of the data has been loaded into the second table, the reporting applications are pointed to the second version of the table and the original version can be dropped; the process continues until all new data has been loaded. A simple way to do this is rename the replacement fact table to the original table's name immediately after dropping the original fact table.

The advantage of this approach is that a table is never being queried while its data is being modified. So, constraints or indexes can be disabled, data can be loaded outside of a transaction, and the statistics can be updated before anyone reads from the table. This idea can be extended to work on the entire schema. With enough space, it is possible to build new versions of all the

fact tables, build new derived tables from them, and then switch everything over at once, for the utmost in data consistency, without affecting availability of the previous snapshot of data.

5.5 Troubleshooting Information for Technical Support

When contacting Technical Support, the following information is very useful:

- Product name, edition, and version (for example, Jaspersoft OLAP Community Project 3.5.1).
In the commercial editions, click **About JasperReports Server** on any page of the application to view these details.
In the Community Edition edition, locate and open the `WEB-INF/internal/jasperserver.properties` file in your JasperReports Server installation; the `JS_VERSION` property indicates your version of the application.
- Operating system, application server, and database name and version.
- A description or screen capture of the issue as it appears in the web application (if applicable).
- Detailed steps to reproduce the problem.
- The text of any error messages found in the `WEB-INF/jasperserver.log` file

If the problem is specific to a particular schema or OLAP view, please also include:

- The OLAP data source definition.
- The OLAP connection definition.
- The schema XML file, or relevant parts.
- The MDX query that illustrates the error.

CHAPTER 6 EXECUTING OLAP VIEWS USING THE HTTP INTERFACE

The MDX HTTP interface facilitates the creation of an OLAP view on the fly via the HTTP protocol. This is helpful when integrating JasperSoft OLAP with other systems, or when you want to pass MDX generated from a report to dynamically generate an OLAP view.

Before using the HTTP interface, insure that your system will meet these requirements:

- JasperReports Server must be installed.
- The OLAP view to open must exist in JasperSoft OLAP.
- The currently logged-in user must have sufficient privileges for the OLAP view.
- The MDX you supply must be valid.

6.1 Syntax

The basic syntax for calling the OLAP view is:

```
http://<server URL>:<port>/<context-path>/olap/viewOlap.html?name=<view name>&mdx=
<MDX query>
```

where:

Variable	Description
<server URL>	Application server URL and port defined in server.xml.
<port>	The application server's port number. For default for Tomcat the is 8080.
<context-path>	JasperReports Server context path defined in context.xml.
<view name>	OLAP view folder path and name in the repository.
<MDX query>	A valid MDX command that returns data in the view.

6.2 Example HTTP Calls for Jaspersoft OLAP



This section describes functionality that can be restricted by the software license for JasperReports Server. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

In addition to the prerequisites listed above, this example assumes that:

- JasperReports Server is running against Tomcat as its application server, and that 8080 is your port.
- The sample data has been installed.
- The Foodmart sample analysis, which is included in the sample data, is available in the analysis/views folder.
- The currently logged in user is the demo user, which is included in the sample data.

The following is a very simple example of the URL to pass to view the Foodmart OLAP view in the commercial editions:

```
http://test15.jaspersoft.com:8080/jasperserver-pro/olap/viewOlap.html?name=/analysis/views/Foodmart_sample
```

The values in the example correspond to the syntax for calling the OLAP view as follows:

Value	Variable
test15.jaspersoft.com	<server URL>
8080	<port>
jasperserver-pro	<context-path>
Foodmart_sample	<view name>

You can also specify MDX to pass, which changes the initial state of the view displayed. The following example displays the Foodmart sample view:

```
http://test15.jaspersoft.com:8080/jasperserver-pro/olap/viewOlap.html?name=/analysis/views/Foodmart_sample&mdx=select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]} ON COLUMNS, Hierarchize([Store].[Store Country].[USA].children) ON ROWS from [Sales]
```

The values in the example correspond to the syntax as follows:

Parameter	Value	Variable
&mdx=	select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]} ON COLUMNS, Hierarchize([Store].[Store Country].[USA].children) ON ROWS from [Sales]	<MDX query>

The URL returns this OLAP view:

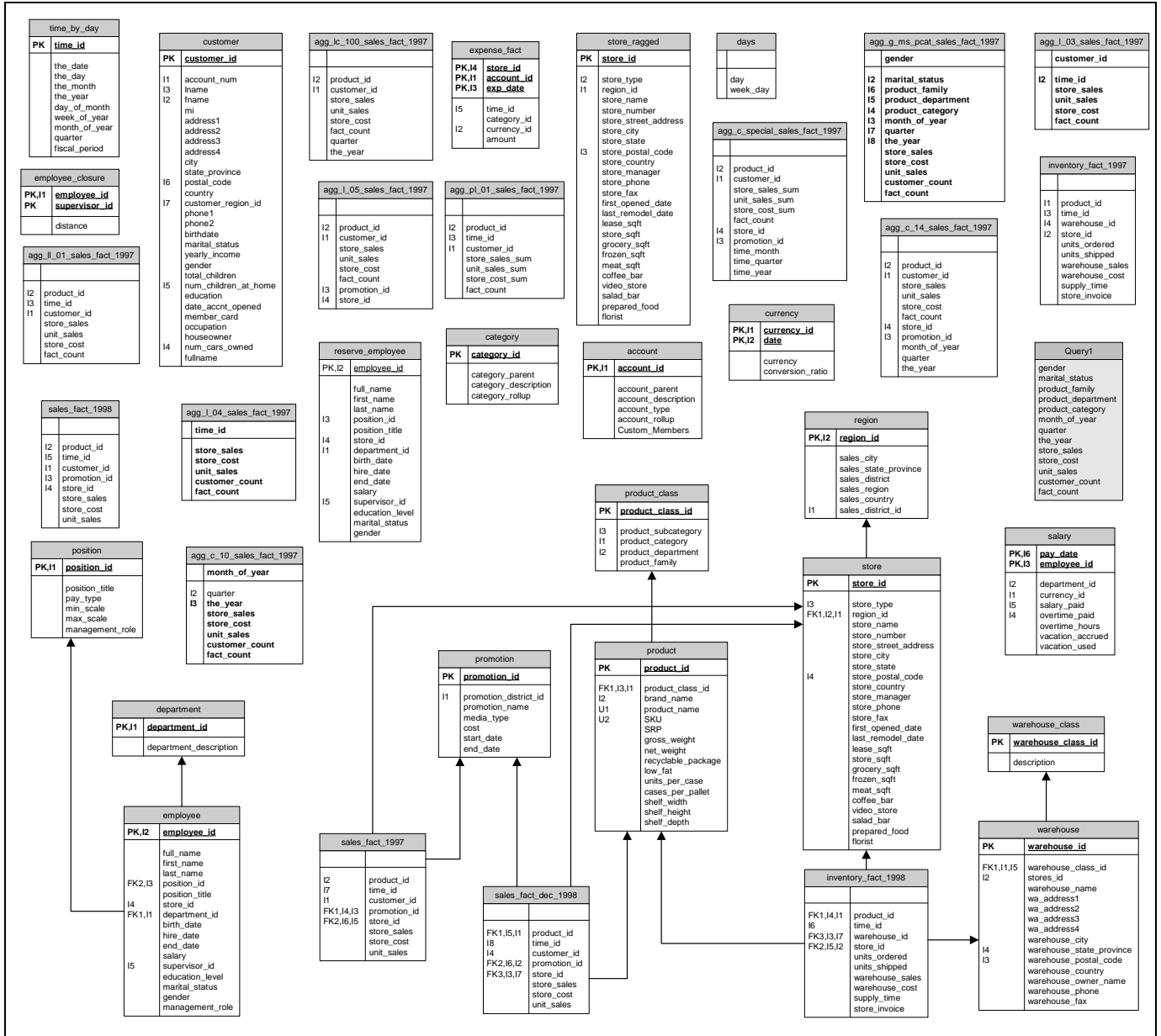
Dimensions			Measures
Store	Store Country	Store State	Unit Sales
All Stores	USA	+ CA	74,748
		+ OR	67,659

Figure 6-1 Foodmart Sample OLAP view Returned by MDX HTTP Interface

APPENDIX A EXAMPLE FOODMART SCHEMAS

This appendix has a chart the Foodmart database schema and the XML code of the schema of a Foodmart OLAP view.

A.1 FoodMart Database Schema



A.2 FoodMart Sales OLAP Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<Schema name="FOODMART">
  <Cube name="SALES BY PRODUCT BY STORE">
    <Dimension name="PRODUCT">
      <Hierarchy hasAll="true" name="PRODUCT CLASS" primaryKey="product_id"
        primaryKeyTable="product">
        <Join leftAlias="product" leftKey="product_class_id" rightAlias="product_class"
          rightKey="product_class_id">
          <Table name="product_class" alias="product_class"/>
          <Table name="product" alias="product"/>
        </Join>
        <Level column="product_family" name="PRODUCT FAMILY" table="product_class"
          uniqueMembers="true"/>
        <Level column="product_department" name="PRODUCT DEPARTMENT" table="product_class"
          uniqueMembers="false"/>
        <Level column="product_category" name="PRODUCT CATEGORY" table="product_class"
          uniqueMembers="false"/>
        <Level column="product_subcategory" name="PRODUCT SUBCATEGORY" table="product_class"
          uniqueMembers="false"/>
        <Level column="brand_name" name="BRAND NAME" table="product"
          uniqueMembers="false"/>
        <Level column="product_name" name="PRODUCT NAME" table="product"
          uniqueMembers="true"/>
      </Hierarchy>
    </Dimension>
    <Dimension name="STORE">
      <Hierarchy hasAll="true" name="STORE LOCATION" primaryKey="store_id">
        <Table name="store"/>
        <Level column="store_country" name="STORE COUNTRY" uniqueMembers="true"/>
        <Level column="store_state" name="STORE STATE" uniqueMembers="true"/>
        <Level column="store_city" name="STORE CITY" uniqueMembers="false"/>
        <Level column="store_name" name="STORE NAME" uniqueMembers="true">
          <Property column="store_type" name="STORE TYPE"/>
          <Property column="store_manager" name="STORE MANAGER"/>
          <Property column="store_sqft" name="STORE SQFT" type="Numeric"/>
          <Property column="grocery_sqft" name="GROCERY SQFT" type="Numeric"/>
          <Property column="frozen_sqft" name="FROZEN SQFT" type="Numeric"/>
          <Property column="meat_sqft" name="MEAT SQFT" type="Numeric"/>
          <Property column="coffee_bar" name="HAS COFFEE BAR" type="Boolean"/>
          <Property column="store_street_address" name="STREET ADDRESS" type="String"/>
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure aggregator="sum" column="store_sales" formatString="$#,###.00"
      name="STORE SALES"/>
    <Measure aggregator="count" column="product_id" formatString="#,###" name="SALES COUNT"/>
  </Cube>
</Schema>

```


GLOSSARY

Ad Hoc Editor

The interactive report designer in JasperReports Server Professional and Enterprise editions. Starting from a collection of fields predefined in a Topic or selected from a Domain, the Ad Hoc Editor lets you drag and drop report elements to draft, preview, and finalize reports. Like JRXML reports, Ad Hoc reports can be run, printed, and scheduled within JasperReports Server. In addition, Ad Hoc reports may be reopened in the Ad Hoc Editor, further modified, and saved.

Analysis Client Connection

A definition for retrieving an analysis view. An analysis client connection is either a direct Java connection (Mondrian connection) or an XML-based API connection (XML/A connection).

Analysis Schema

A metadata definition of a multidimensional database. In Jaspersoft OLAP, schemas are stored in the repository as XML file resources.

Analysis View

A view of multidimensional data that is based on an analysis client connection and an MDX query. It is the entry point to analysis operations, such as slice and dice, drill down, and drill through.

Audit Archiving

To prevent audit logs from growing too large to be easily accessed, the system installer configures JasperReports Server to move current audit logs to an archive after a certain number of days, and to delete logs in the archive after a certain age. The archive is another table in the JasperReports Server's private database.

Audit Domains

A Domain that accesses audit data in the repository and lets administrators create Ad Hoc reports of server activity. There is one Domain for current audit logs and one for archived logs.

Audit Logging

When auditing is enabled, audit logging is the active recording of who used JasperReports Server to do what when. The system installer can configure what activities to log, the amount of detail gathered, and when to archive the data. Audit logs are stored in the same private database that JasperReports Server uses to store the repository, but the data is only accessible through the audit Domains.

Auditing

A feature of JasperReports Server Enterprise edition that records all server activity and allows administrators to view the data.

Calculated Field

In a Domain, a field whose value is calculated from a user-written formula that may include any number of fields, operators, and constants. A calculated field is defined in the Domain Designer, and it becomes one of the items to which the Domain's security file and locale bundles can apply.

CRM

Customer Relationship Management. The practice of managing every facet of a company's interactions with its clientele. CRM applications help businesses track and support their customers.

CrossJoin

An MDX function that combines two or more dimensions into a single axis (column or row).

Cube

The basis of most analysis applications, a cube is a data structure that contains three or more dimensions that categorize the cube's quantitative data. When you navigate the data displayed in an analysis view, you are exploring a cube.

Custom Field

In the Ad Hoc Editor, a field that is created through menu items as a simple function of one or two available fields, including other custom fields. When a custom field becomes too complex or needs to be used in many reports, it is best to define it as a calculated field in a Domain.

Dashboard

A collection of reports, input controls, graphics, labels, and web content displayed in a single, integrated view. Dashboards often present a high level view of your data, but input controls can parameterize the data to display. For example, you can narrow down the data to a specific date range. Embedded web content, such as other web-based applications or maps, make dashboards more interactive and functional.

Derived Table

In a Domain, a derived table is defined by an additional query whose result becomes another set of items available in the Domain. For example, with a JDBC data source, you can write an SQL query that includes complex functions for selecting data. You can use the items in a derived table for other operations on the Domain, such as joining tables, defining a calculated field, or filtering. The items in a derived table can also be referenced in the Domain's security file and locale bundles.

Data Policy

In JasperReports Server, a setting that determines how the server processes and caches data used by Ad Hoc reports. Select your data policies by clicking **Manage > Ad Hoc Settings**.

Data Source

Defines the connection properties that JasperReports Server needs to access data. The server transmits queries to data sources and obtains datasets in return for use in filling reports and previewing Ad Hoc reports. JasperReports Server supports JDBC, JNDI, and Bean data sources; custom data sources can be defined as well.

Dataset

A collection of data arranged in columns and rows. Datasets are equivalent to relational results sets and the `JRDataSource` type in the JasperReports Library.

Datatype

In JasperReports Server, a datatype is used to characterize a value entered through an input control. A datatype must be of type text, number, date, or date-time. It can include constraints on the value of the input, for example maximum and minimum values. As such, a datatype in JasperReports Server is more structured than a datatype in most programming languages.

Denormalize

A process for creating table joins that speeds up data retrieval at the cost of having duplicate row values between some columns.

Dice

An OLAP operation to select columns.

Dimension

A categorization of the data in a cube. For example, a cube that stores data about sales figures might include dimensions such as time, product, region, and customer's industry.

Domain

A virtual view of a data source that presents the data in business terms, allows for localization, and provides data-level security. A Domain is not a view of the database in relational terms, but it implements the same functionality within JasperReports Server. The design of a Domain specifies tables in the database, join clauses, calculated fields, display names, and default properties, all of which define items and sets of items for creating Ad Hoc reports.

Domain Topic

A Topic that is created from a Domain by the Data Chooser. A Domain Topic is based on the data source and items in a Domain, but it allows further filtering, user input, and selection of items. Unlike a JRXML-based Topic, a Domain Topic can be edited in JasperReports Server by users with the appropriate permissions.

Drill

To click on an element of an analysis view to change the data that is displayed:

- **Drill down.** An OLAP operation that exposes more detailed information down the hierarchy levels by delving deeper into the hierarchy and updating the contents of the navigation table.
- **Drill through.** An OLAP operation that displays detailed transactional data for a given aggregate measure. Click a fact to open a new table beneath the main navigation table; the new table displays the low-level data that constitutes the data that was clicked.
- **Drill up.** An OLAP operation for returning the parent hierarchy level to view to summary information.

Eclipse

An open source Integrated Development Environment (IDE) for Java and other programming languages, such as C/C++.

ETL

Extract, Transform, Load. A process that retrieves data from transactional systems, and filters and aggregates the data to create a multidimensional database. Generally, ETL prepares the database that your reports will access. The Jaspersoft ETL product lets you define and schedule ETL processes.

Fact

The specific value or aggregate value of a measure for a particular member of a dimension. Facts are typically numeric.

Field

A field is equivalent to a column in the relational database model. Fields originate in the structure of the data source, but you may define calculated fields in a Domain or custom fields in the Ad Hoc Editor. Any type of field, along with its display name and default formatting properties, is called an item and may be used in the Ad Hoc Editor.

Frame

A dashboard element that displays reports or custom URLs. Frames can be mapped to input controls if their content can accept parameters.

Group

In a report, a group is a set of data rows that have an identical value in a designated field.

- In a table, the value appears in a header and footer around the rows of the group, while the other fields appear as columns.
- In a chart, the field chosen to define the group becomes the independent variable on the X axis, while the other fields of each group are used to compute the dependent value on the Y axis.

Hierarchy Level

In analysis, a member of a dimension containing a group of members.

Input Control

A button, check box, drop-down list, text field, or calendar icon that allows users to enter a value when running a report or viewing a dashboard that accepts input parameters. For JRXML reports, input controls and their associated datatypes must be defined as repository objects and explicitly associated with the report. For Domain-based reports that prompt for filter values, the input controls are defined internally. When either type of report is used in a dashboard, its input controls are available to be added as special content.

iReport Designer

An open source tool for graphically designing reports that leverage all features of the JasperReports Library. The Jaspersoft iReport Designer lets you drag and drop fields, charts, and sub-reports into a canvas, and also define parameters or expressions for each object to create pixel-perfect reports. iReport Designer outputs the JRXML of the report or uploads it directly to JasperReports Server.

Item

When designing a Domain or creating a Topic based on a Domain, an item is the representation of a database field or a calculated field along with its display name and formatting properties defined in the Domain. Items can be grouped in sets and are available for use in the creation of Ad Hoc reports.

JasperReports Library

An embeddable, open source, Java API for generating a report, filling it with current data, drawing charts and tables, and exporting to any standard format (HTML, PDF, Excel, CSV, and others). JasperReports processes reports defined in JRXML, an open XML format that allows the report to contain expressions and logic to control report output based on run-time data.

JasperReports Server

A commercial open source, server-based application that calls the JasperReports library to generate and share reports securely. JasperReports Server authenticates users and lets them upload, run, view, schedule, and send reports from a web browser. Commercial versions provide metadata layers, interactive report and dashboard creation, and enterprise features such as organizations and auditing.

Jaspersoft ETL

A graphical tool for designing and implementing your data extraction, transforming, and loading (ETL) tasks. It provides hundreds of data source connectors to extract data from many relational and non-relational systems. Then, it schedules and performs data aggregation and integration into data marts or data warehouses that you use for reporting.

Jaspersoft OLAP

A relational OLAP server integrated into JasperReports Server that performs data analysis with MDX queries. The product includes query builders and visualization clients that help users explore and make sense of multidimensional data. Jaspersoft OLAP also supports XML/A connections to remote servers.

JavaBean

A reusable Java component that can be dropped into an application container to provide standard functionality.

JDBC

Java Database Connectivity. A standard interface that Java applications use to access databases.

JNDI

Java Naming and Directory Interface. A standard interface that Java applications use to access naming and directory services.

Join Tree

In Domains, a collection of joined tables from the actual data source. A join is the relational operation that associates the rows of one table with the rows of another table based on a common value in given field of each table. Only the fields in a same join tree or calculated from the fields in a same join tree may appear together in a report.

JPivot

An open source graphical user interface for OLAP operations. For more information, visit <http://jpivot.sourceforge.net/>.

JRXML

An XML file format for saving and sharing reports created for the JasperReports Library and the applications that use it, such as iReport Designer and JasperReports Server. JRXML is an open format that uses the XML standard to define precisely all the structure and configuration of a report.

MDX

Multidimensional Expression Language. A language for querying multidimensional objects, such as OLAP (On Line Analytical Processing) cubes, and returning cube data for analytical processing. An MDX query is the query that determines the data displayed in an analysis view.

Measure

Depending on the context:

- In a report, a formula that calculates the values displayed in a table's columns, a crosstab's data values, or a chart's dependent variable (such as the slices in a pie).
- In an analysis view, a formula that calculates the facts that constitute the quantitative data in a cube.

Mondrian

A Java-based, open source multidimensional database application.

Mondrian Connection

An analysis client connection that consists of an analysis schema and a data source used to populate an analysis view.

Mondrian Schema Editor

An open source Eclipse plugin for creating Mondrian analysis schemas.

Mondrian XMLA Source

A server-side XMLA source definition of a remote client-side XML/A connection used to populate an analysis view using the XMLA standard.

MySQL

An open source relational database management system. For information, visit <http://www.mysql.com/>.

Navigation Table

The main table in an analysis view that displays measures and dimensions as columns and rows.

ODBO Connect

Jaspersoft ODBO Connect enables Microsoft Excel 2003 and 2007 Pivot Tables to work with Jaspersoft OLAP and other OLAP servers that support the XML/A protocol. After setting up the Jaspersoft ODBO data source, business analysts can use Excel Pivot Tables as a front-end for OLAP analysis.

OLAP

On Line Analytical Processing. Provides multidimensional views of data that help users analyze current and past performance and model future scenarios.

Organization

A set of users that share folders and resources in the repository. An organization has its own user accounts, roles, and root folder in the repository to securely isolate it from other organizations that may be hosted on the same instance of JasperReports Server.

Organization Admin

Also called the organization administrator. A user in an organization with the privileges to manage the organization's user accounts and roles, repository permissions, and repository content. An organization admin can also create sub-organizations and manage all of their accounts, roles, and repository objects. The default organization admin in each organization is the `jasperadmin` account.


Outlier

A fact that seems incongruous when compared to other member's facts. For example, a very low sales figure or a very high number of helpdesk tickets. Such outliers may indicate a problem (or an important achievement) in your business. The analysis features of Jaspersoft OLAP excel at revealing outliers.

Parameter

Named values that are passed to the engine at report-filling time to control the data returned or the appearance and formatting of the report. A report parameter is defined by its name and type. In JasperReports Server, parameters can be mapped to input controls that users can interact with.

Pivot

To rotate a crosstab such that its row groups become column groups and its column groups become rows. In the Ad Hoc Editor, pivot a crosstab by clicking .

Pivot Table

A table with two physical dimensions (for example, X and Y axis) for organizing information containing more than two logical dimensions (for example, PRODUCT, CUSTOMER, TIME, and LOCATION), such that each physical dimension is capable of representing one or more logical dimensions, where the values described by the dimensions are aggregated using a function such as SUM. Pivot tables are used in Jaspersoft OLAP.

Properties

Settings associated with an object. The settings determine certain features of the object, such as its color and label. Properties are normally editable. In Java, properties can be set in files listing objects and their settings.

Repository

The tree structure of folders that contain all saved reports, dashboards, analysis views, and resources. Users access the repository through the JasperReports Server web interface or through iReport. Applications can access the repository through the web service API. Administrators use the import and export utilities to back up the repository contents.

Resource

In JasperReports Server, anything residing in the repository, such as an image, file, font, data source, Topic, Domain, report element, saved report, report output, dashboard, or analysis view. Resources also include the folders in the repository. Administrators set user and role-based access permissions on repository resources to establish a security policy.

Role

A security feature of JasperReports Server. Administrators create named roles, assign them to user accounts, and then set access permissions to repository objects based on those roles. Certain roles also determine what functionality and menu options are displayed to users in the JasperReports Server interface.

Schema

A logical model that determines how data is stored. For example, the schema in a relational database is a description of the relationships between tables, views, and indexes. In Jaspersoft OLAP, an OLAP schema is the logical model of the data that appears in an analysis view; they are uploaded to the repository as resources. For Domains, schemas are represented in XML design files.

Schema Workbench

A graphical tool for easily designing OLAP schemas, data security schemas, and MDX queries. The resulting cube and query definitions can then be used in Jaspersoft OLAP to perform simple but powerful analysis of large quantities of multi-dimensional data stored in standard RDBMS systems.

Set

In Domains and Domain Topics, a named collection of items grouped together for ease of use in the Ad Hoc Editor. A set can be based on the fields in a table or entirely defined by the Domain creator, but all items in a set must originate in the same join tree. The order of items in a set is preserved.

Slice

An OLAP operation for filtering data rows.

SQL

Structured Query Language. A standard language used to access and manipulate data and schemas in a relational database.

System Admin

Also called the system administrator. A user who has unlimited access to manage all organizations, users, roles, repository permissions, and repository objects across the entire JasperReports Server instance. The system admin can create root-level organizations and manage all server settings. The default system admin is the `superuser` account.

Topic

A JRXML file created externally and uploaded to JasperReports Server as a basis for Ad Hoc reports. Topics are created by business analysts to specify a data source and a list of fields with which business users can create reports in the Ad Hoc Editor. Topics are stored in the Ad Hoc Components folder of the repository and displayed when a user launches the Ad Hoc Editor.

Transactional Data

Data that describe measurable aspects of an event, such as a retail transaction, relevant to your business. Transactional data are often stored in relational databases, with one row for each event and a table column or field for each measure.

User

Depending on the context:

- A person who interacts with JasperReports Server through the web interface. There are generally three categories of users: administrators who install and configure JasperReports Server, database experts or business analysts who create data sources and Domains, and business users who create and view reports and dashboards.
- A user account that has an ID and password to enforce authentication. Both people and API calls accessing the server must provide the ID and password of a valid user account. Roles are assigned to user accounts to determine access to objects in the repository.

WCF

Web Component Framework. A low-level GUI component of JPivot. For more information, see <http://jpivot.sourceforge.net/wcf/index.html>.

Web Services

A SOAP (Simple Object Access Protocol) API that enables applications to access certain features of JasperReports Server. The features include repository, scheduling and user administration tasks.

XML

eXtensible Markup language. A standard for defining, transferring, and interpreting data for use across any number of XML-enabled applications.

XML/A

XML for Analysis. An XML standard that uses Simple Object Access protocol (SOAP) to access remote data sources. For more information, see <http://www.xmla.org/>

XML/A Connection

A type of analysis client connection that consists of Simple Object Access Protocol (SOAP) definitions used to populate an analysis view.

INDEX

A

access grant definitions 24, 65, 75
 access roles 68
 administering Jaspersoft OLAP
 bottleneck analysis 90
 data security 63
 integrating data flow 96
 knowledge required 77, 85, 92
 maintenance tasks 88
 profiling 91
 system maintenance 85
 testing data access 72
 troubleshooting system performance 88
 tuning database performance 92
 tuning system performance 90
 AGXML 65, 75
 Amdahl's law 91
 analysis views. *See* OLAP views.
 analyzing data
 in Jaspersoft OLAP 28
 tools for 39

B

bottleneck analysis 90

C

cache
 and load balancing 90
 clearing the cache 89
 comparison with JasperReports Server caching 89
 integrating data flow 97
 pre-caching 90
 charts
 in navigation tables 51
 in OLAP views 37
 columns
 See also dimension columns.
 configuring measures 41

 hiding 48
 commercial editions
 added features 7
 administering 63
 data security 63
 documentation 8
 Community Project 7, 8
 configuring
 data cubes 40
 data security 63–76
 dimensions 43, 45
 filters 43, 45
 measures 41
 navigation tables 39
 connections
 in OLAP views 15, 27
 Mondrian 20, 27
 Mondrian connections vs. XML/A 91
 OLAP client 20, 86
 XML/A 27, 86
 constructs
 FROM 45
 SELECT 45
 WHERE 45
 creating
 data sources 15
 MySQL database 15
 OLAP views 27
 OLAP views with XML/A 87
 cubes
 and access to data 63
 configuring 40
 cube content 40
 database structure 18
 design concepts 77
 dimensions 12
 facts 13

- in dimensional modeling 77
- measures 13
- overview 12
- sorting 45

D

- data security 63
 - data sources
 - and OLAP views 15, 17
 - data preparation 14
 - data warehouse 96
 - error messages 57
 - importing data 16
 - MySQL database 15
 - overview 17
 - preparing data for analysis 15
 - data warehouse 96
 - database administrator
 - data security 63
 - example schemas 101
 - guide to using Ultimate Guide 8
 - integrating data flow 96
 - knowledge required 92
 - tuning database performance 92
 - database, cubes built from 18
 - design concepts 77
 - dimension columns 47
 - dimensional modeling
 - creating processes 79
 - example 79
 - overview 77
 - dimensions
 - as filters 43, 45
 - configuring 48
 - defined 12
 - example 67
 - in columns 42
 - in dimensional modeling 77
 - swapping with measures 48
 - documentation
 - guide to using Ultimate Guide 8
 - premium documentation 9
 - standard documentation 8
 - drill-through table functions
 - Edit Properties pane 51
 - in OLAP views 36
 - navigation controls 49
- ## E
- editions of Jaspersoft OLAP 7
 - Enterprise edition. *See* commercial editions.
 - error messages
 - common messages 55
 - data sources 57
 - Jaspersoft OLAP 54
 - Mondrian schemas 57

- OLAP 57–62
- OLAP views 56
- resources 57–62

examples

- access grant definition 75
 - adding dimensions 81
 - AGXML 75
 - analyzing data 28
 - changing layouts 82
 - data security 63
 - dimensional modeling 79
 - dimensions 67
 - displaying charts 37, 83
 - displaying OLAP views 31, 34
 - drill-through table functions 36, 80, 84
 - Excel output 38
 - implementation 63
 - MDX statements 72, 79, 83
 - measure 67
 - OLAP schemas 68, 74, 102
 - OLAP views 72, 79, 87
 - pivoting 82
 - profiling 91
 - schemas 68, 74, 102, 103
 - slice and dice 82
 - sorting OLAP views 35
 - XML/A definition 85
 - zoom functions 80
- Excel spreadsheet output 53
- exporting OLAP views 38

F

- facts 12, 13
- filters 43, 45
- Foodmart
 - database 15, 16, 18, 79, 102
 - OLAP schema 22, 103
 - OLAP view 28, 29, 79
- FROM construct 45

H

- HTTP interface to Jaspersoft OLAP 99

I

- importing data 16

J

- JasperReports 7, 9, 91
- JasperReports Server
 - adding data sources 17
 - and Jaspersoft OLAP 13
 - repository 13
- Jaspersoft 7
- Jaspersoft BI Suite 9
- Jaspersoft OLAP
 - and JasperReports Server 13
 - business intelligence platform 7
 - creating OLAP views 27

- data preparation 14
- design concepts 77
- differences between editions 7, 39, 91
- editions 7
- error messages 54
- HTTP interface 99
- measures 41
- output 53
- overview 11, 13, 14, 27
- profiling 91
- technical support 98
- tools 39, 48
- troubleshooting 54, 98
- user interface 39
- versions 14

Jaspersoft OLAP Workbench 12, 68

JPivot 7

L

- load balancing
 - and clearing the cache 90
 - in large systems 95

M

MDX queries. *See* queries.

measures

- configuring 41, 48
- example 67
- in cubes 13
- in dimensional modeling 77
- in schemas 12
- swapping with dimensions 48

member grants 65

modeling

- access to data 63
- overview 77

Mondrian

- about 7
- and OLAP views 19, 20
- error messages 57
- Mondrian connection 71
- schemas. *See* OLAP schemas.

Mondrian connections and OLAP views 20

MySQL and data security example 64

N

navigation controls

- drill-through table functions 49
- Edit Properties pane 51
- Expand Member 48
- Expand Position 48
- zoom on drill 49

navigation tables

- charts 51
- configuring 39
- controls 48
- output 53

O

OLAP

- adding a client connection 20
- adding a schema 18
- connections 20, 71, 86
- cubes 12
- error messages 57–62
- integrating data flow 97
- Mondrian schemas 12
- OLAP engine 95
- operations 15
- overview 11
- schemas 12

OLAP views

- charts 37
- connections 15
- creating 27
- cube content 40
- data sources 15, 17
- databases 15
- defined 27
- displaying examples 31
- drill-through 36
- error messages 56
- example 31, 72, 87
- exporting output 38
- HTTP interface 99
- importing data 16
- MDX queries 24
- Mondrian connections 19, 20, 27
- OLAP connections 20
- OLAP schemas 18, 19
- queries 44
- resources 15
- sample 31
- saving 54
- schemas 15
- sorting 35
- testing data access 72
- XML/A 27, 85, 87

open source 9

overviews

- cubes 12
- data security 63
- dimensional modeling 77
- Jaspersoft OLAP 11, 13, 27
- Jaspersoft OLAP design concepts 77
- modeling 77
- OLAP 11
- XML/A 85

P

PDF file output 53

pivoting 82

Professional edition. *See* commercial editions.

profile attributes 65, 70

Q

queries

- adding an MDX query 30
- configuring data cubes 44, 45
- contents of MDX query 45
- editing 45
- in OLAP views 24
- MDX 24, 30
- passed via HTTP 100
- query editor 44
- sample MDX query 24, 72, 80
- scope 45

query editor 44

R

resources

- error messages 57–62
- in OLAP views 15

roles 68

rows

- hiding 48
- row level security 63

S

saving views 54

schemas

- database schemas 18
- example 68, 74
- FoodMart database schema 102
- FoodMart OLAP schema 103
- in OLAP views 15
- Mondrian. *See* OLAP schemas.
- OLAP 12, 18, 19
- optimizing the schema design 94
- tuning the database 94

securing data 63

SELECT construct 45

sorting

- data cubes 45
- OLAP views 35

substituting variables in AGXML 65

system administrator

- bottleneck analysis 90, 92
- example schemas 101
- guide to using Ultimate Guide 8
- integrating data flow 96
- knowledge required 77, 85, 92
- maintenance tasks 85, 88
- monitoring system performance 88
- profiling Jaspersoft OLAP 91
- securing data 63
- troubleshooting system performance 88, 98
- tuning database performance 92
- tuning system performance 90

system developer

- example schemas 101

guide to using Ultimate Guide 8

integrating data flow 96

securing data 63

T

technical business analyst

- creating views 27
- guide to using Ultimate Guide 8
- integrating data flow 96
- securing data 63

technical support 98

tool bars 39

tools

- charts 51
- data cubes 40
- dimension columns 47
- exporting output 53
- navigation table controls 48
- tool bar 39, 40

trickle and flip 97

troubleshooting

- common errors 55
- data sources 57
- Jaspersoft OLAP 54
- Mondrian schemas 57
- OLAP 57–62
- OLAP views 56
- resources 57–62
- system performance 88, 98

U

user guides. *See* documentation.

user interface

- columns 41
- dimension columns 47
- dimensions 42, 43
- filters 43
- measures 41
- navigation tables 39
- tool bars 39

user types

- database administrator 8
- system administrator 8
- system developer 8
- technical business analyst 8

users 69

V

variable substitution 65, 70

versions of Jaspersoft OLAP 7

W

WHERE construct 45

Workbench. *See* Jaspersoft OLAP Workbench.

X

XML 12

XML/A

and load balancing 91
and OLAP views 87
connections 86

functional definition 85
overview 85
vs. Mondrian connections 91

