



# JasperReports®

## OLAP Ultimate Guide

Version 9.0.0 | January 2024

# Contents

---

<b>Contents</b> .....	<b>2</b>
<b>Introduction to Jaspersoft® OLAP</b> .....	<b>5</b>
Community and Commercial Editions .....	7
User Descriptions and Document Maps .....	7
Technical Business Analyst .....	7
System Developer .....	7
System Administrator and Database Administrator .....	8
<b>Getting Started</b> .....	<b>9</b>
On-Line Analytical Processing .....	9
OLAP Schema .....	10
Cubes .....	11
Dimensions .....	11
Measures and Facts .....	12
Jaspersoft OLAP and the Repository .....	13
Creating a Jaspersoft OLAP Environment .....	14
Creating the Database .....	15
Defining a Data Source in JasperReports Server .....	17
Creating and Uploading an OLAP Schema .....	17
Creating a Mondrian Connection .....	17
Defining an MDX Query .....	18
Configuring XML/A .....	18
<b>Analyzing Data in a View</b> .....	<b>20</b>
OLAP Views .....	20
Overview of OLAP View Analysis .....	21
Displaying Product Sales by Quarter .....	21

Displaying Product Sales by Time Across Store Locations .....	27
Charts and Exporting .....	28
Using the OLAP View Tools .....	28
Analysis Tool Bar and Navigation Table .....	28
Cube Configuration .....	30
Dimension Column Layout .....	41
Navigation Table Controls .....	44
<b>Securing Data in Jaspersoft OLAP .....</b>	<b>49</b>
Securing Jaspersoft OLAP Data: A Business Case .....	50
Overview of CZS's Process .....	50
Understanding Access Grant Definitions and Attributes .....	52
About Access Grant Definitions .....	52
Attributes and Variable Substitution .....	54
Configuring CZS's Ad Hoc View .....	55
Defining a Sales Numbers Cube .....	55
Creating an OLAP Schema .....	56
Determining Roles .....	57
Selecting Users for the Roles .....	58
Assigning Attributes .....	59
Creating a Mondrian Connection .....	62
Creating a Sales Numbers Ad Hoc View .....	62
Testing the Results .....	63
Reference Material .....	66
OLAP Schema .....	66
Access Grant Definition for CZS .....	67
<b>Administering Jaspersoft OLAP .....</b>	<b>69</b>
Understanding the Design Concepts .....	69
Introduction to Dimensional Modeling .....	69
Designing the Model .....	70
Applying the Model .....	73

Maintaining the System .....	79
XML/A Definitions .....	80
Monitoring the System .....	81
Maintaining Tables .....	82
Clearing the Cache .....	83
Tuning Performance .....	83
Understand Jaspersoft OLAP Performance .....	83
Profiling Performance .....	86
Jaspersoft OLAP Tuning Process and Options .....	88
Integrating Jaspersoft OLAP in the Enterprise’s Data Flow .....	95
Troubleshooting Information for Technical Support .....	98
<b>Glossary .....</b>	<b>100</b>
<b>Jaspersoft Documentation and Support Services .....</b>	<b>115</b>
<b>Legal and Third-Party Notices .....</b>	<b>117</b>

# Introduction to Jaspersoft® OLAP

---

JasperReports® Server builds on JasperReports® Library as a comprehensive family of Business Intelligence (BI) products, providing robust static and interactive reporting, report server, and data analysis capabilities. These capabilities are available as either stand-alone products, or as part of an integrated end-to-end BI suite utilizing common metadata and provide shared services, such as security, a repository, and scheduling. The server exposes comprehensive public interfaces enabling seamless integration with other applications and the capability to easily add custom functionality.



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.

The heart of the Jaspersoft® BI Suite is the server, which provides the ability to:

- Easily create new reports based on views designed in an intuitive, web-based, drag and drop Ad Hoc Editor.
- Efficiently and securely manage many reports.
- Interact with reports, including sorting, changing formatting, entering parameters, and drilling on data.
- Schedule reports for distribution through email and storage in the repository.
- Arrange reports and web content to create appealing, data-rich Jaspersoft Dashboards that quickly convey business trends.

For users interested in multi-dimensional modeling, we offer Jaspersoft® OLAP, which runs as part of the server.

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.

You can use the following sources of information to learn about JasperReports Server:

- Our core documentation describes how to install, administer, and use JasperReports Server and Jaspersoft Studio. Core documentation is available as PDFs in the doc subdirectory of your JasperReports Server installation. You can also access PDF and HTML versions of these guides online from the [Documentation section](#) of the Jaspersoft Community website.
- Our Ultimate Guides document advanced features and configuration. They also include best practice recommendations and numerous examples. You can access PDF and HTML versions of these guides online from the [Documentation section](#) of the Jaspersoft Community website.
- Our [Online Learning Portal](#) lets you learn at your own pace, and covers topics for developers, system administrators, business users, and data integration users. The Portal is available online from the Professional Services section of our [website](#).
- Our free samples, which are installed with JasperReports Library, Jaspersoft Studio, and JasperReports Server, are available and documented online. Please visit our [GitHub repository](#).
- If you have a subscription to our professional support offerings, please contact our Technical Support team when you have questions or run into difficulties. They are available on the web at <https://www.jaspersoft.com/support>.

JasperReports Server is a component of both a community project and commercial offerings. Each integrates the standard features such as security, scheduling, a web services interface, and much more for running and sharing reports. Commercial editions provide additional features, including Ad Hoc views and reports, advanced charts, dashboards, Domains, auditing, and a multi-organization architecture for hosting large BI deployments.

This chapter contains the following sections:

- [Community and Commercial Editions](#)
- [User Descriptions and Document Maps](#)

# Community and Commercial Editions

Jaspersoft OLAP is a component of our community project and commercial offerings. Jaspersoft integrates the standard features of JPivot and Mondrian with additional features, including an enhanced user interface, streamlined tool bars and icons, fast expand and 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.

# User Descriptions and Document Maps

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.

## Technical Business Analyst

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

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

- [On-Line Analytical Processing](#)
- [Analyzing Data in a View](#)
- [Securing Data in Jaspersoft OLAP](#)
- [Understanding the Design Concepts](#)

## System Developer

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

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

- [On-Line Analytical Processing](#)
- [Analyzing Data in a View](#)
- [Integrating Jaspersoft OLAP in the Enterprise's Data Flow](#)

## 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 are a System Administrator or DBA, refer to the following sections of this document:

- [Getting Started](#)
- [Analyzing Data in a View](#)
- [Administering Jaspersoft OLAP](#)



# Getting Started

---

This chapter is an overview of the concepts underlying Jaspersoft OLAP. It is meant to illustrate the high-level steps to take when implementing Jaspersoft OLAP. It also explains how to prepare data for analysis. For information about logging in to JasperReports Server and accessing Jaspersoft OLAP, refer to the Jaspersoft OLAP User Guide.

The chapter has these sections:

- [On-Line Analytical Processing](#)
- [Jaspersoft OLAP and the Repository](#)
- [Creating a Jaspersoft OLAP Environment](#)

## 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 are sold, 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 [Using the OLAP View Tools](#).



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.

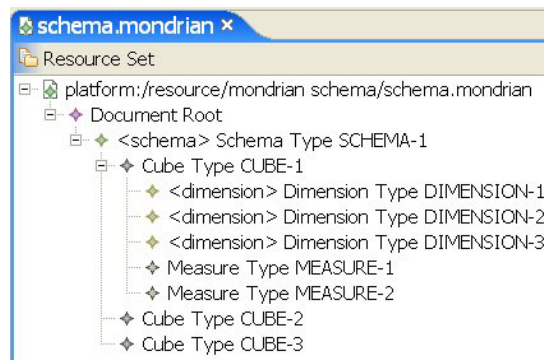
## OLAP Schema

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 the sample schemas in your JasperReports Server installation under `js-install>\samples\OLAP\schemas\FoodmartSchema.xml`).

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



**Figure 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/>.

## 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.

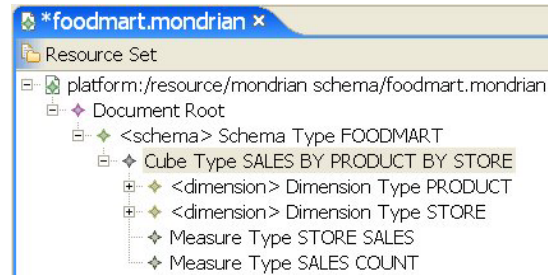
For example, [Jaspersoft OLAP Workbench - Store Cube](#) depicts a cube for sales by product and by store. The cube presents the sales figures (quantitative data) for a range of products and a group of stores (characteristics). The sales figures, such as STORE SALES and SALES COUNT, are the measures, and the product categories and store locations are the dimensions.

## 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, [Hierarchy for the FoodMart Cube](#) 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.

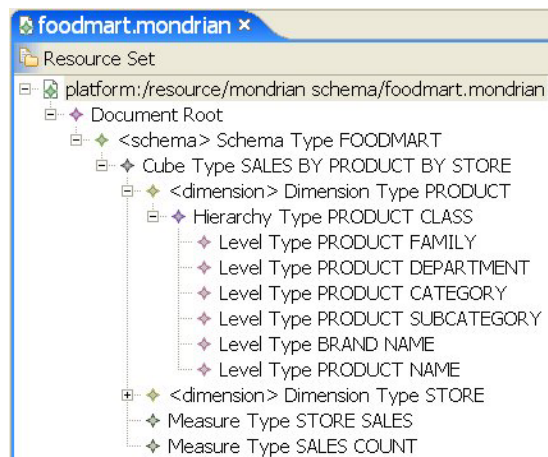


**Figure 2: Hierarchy for the FoodMart Cube**

## 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 [Measure Created with the SUM Aggregation Function](#), STORE SALES is the SUM of the store\_sales column in the transactional data.



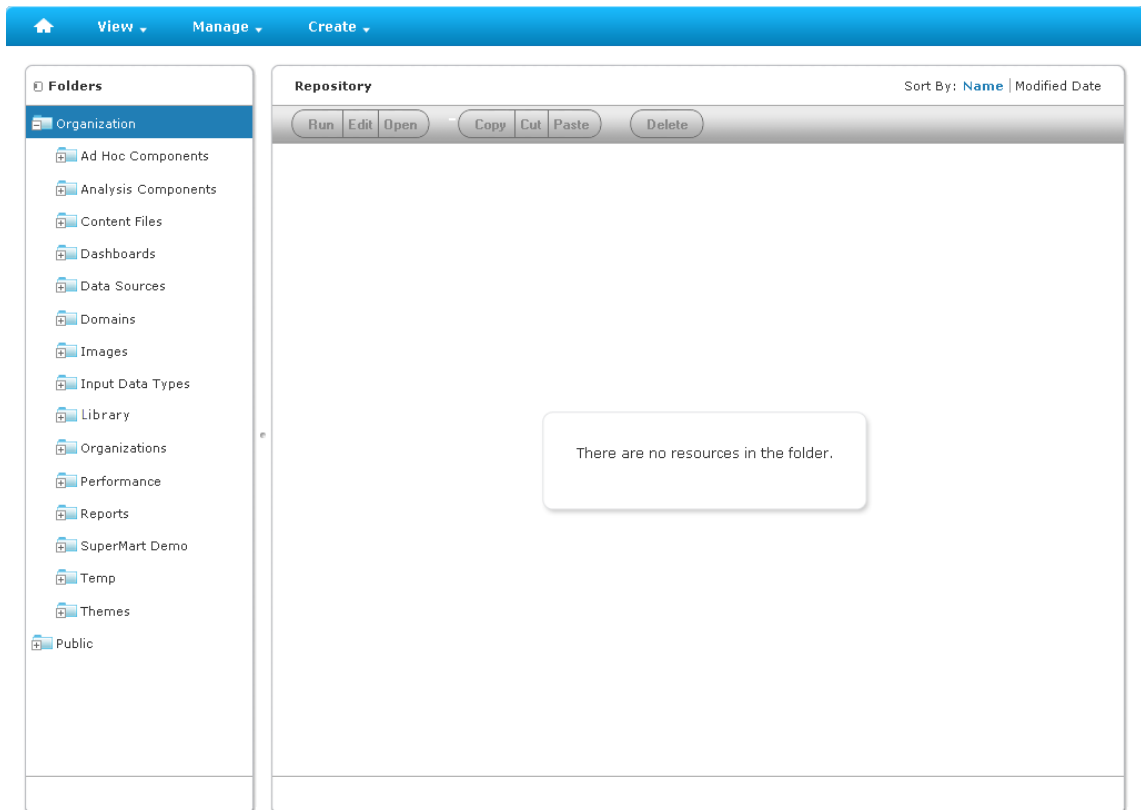
**Figure 3: Measure Created with the SUM Aggregation Function**

# Jaspersoft OLAP and the Repository

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

- Browser-based OLAP interactive views are called OLAP views.
- OLAP objects are 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 Ad Hoc views and reports, 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 Microsoft Excel Pivot Tables to cubes through XML/A. You can get ODBO Connect from <https://community.jaspersoft.com/wiki/getting-started-odbo-connect>.
- Jaspersoft OLAP commercial editions provide the following:
  - OLAP data-level security based on user profiles.
  - Browser-based management of run time parameters.
  - Access to remote XML/A providers from within JasperReports Server.
  - Jaspersoft OLAP performance monitoring.
  - The Ad Hoc Editor can access OLAP client connections directly, and topics can be based on them as well.

In the JasperReports Server repository ([Repository with Default Folders](#)), 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 4: Repository with Default Folders**

For more information on folders and the repository, refer to the Jaspersoft OLAP User Guide.

## Creating a Jaspersoft OLAP Environment

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



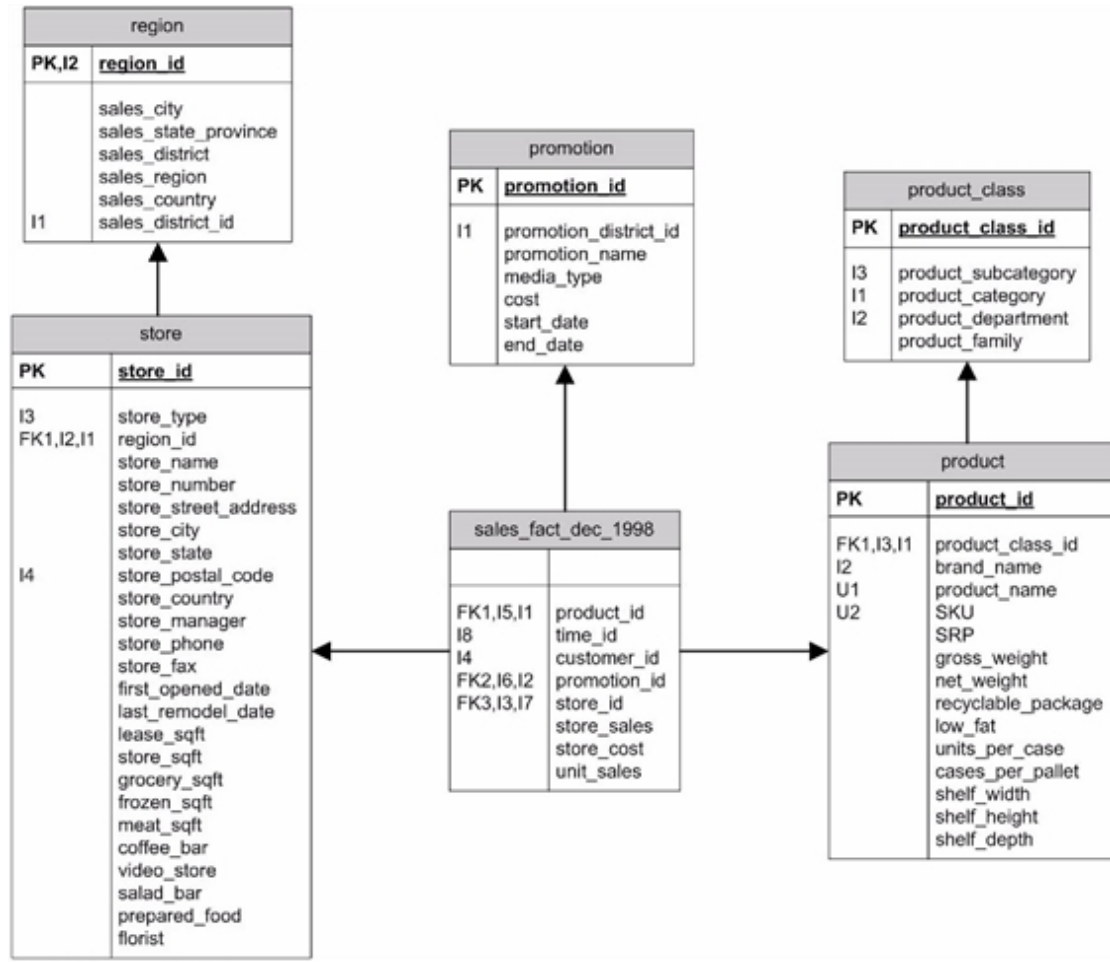
If you installed from the installer and chose the sample data option which Jaspersoft OLAP includes fully prepared sample data, including the FoodMart OLAP view sample described in this section. The installation and configuration steps in this sections are not necessary if you installed the sample data.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

## 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 does not 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, generated by their point-of-sale system, are organized into a multidimensional database that is denormalized to optimize for retrieval and for OLAP operations. An example of a multidimensional database structure is shown in [Database Structure of FoodMart Data Source](#). For a complete example, refer to the sample schemas in your JasperReports Server installation under `js-install>\samples\schemas\FoodmartSchema.xml`.

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



**Figure 5: Database Structure of FoodMart Data Source**

This sample data source is organized in a data structure called a 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 [Database Structure of FoodMart Data Source](#) is included with JasperSoft OLAP as sample data that can be installed with the product.



## Defining a Data Source in JasperReports Server

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

A data source is a connection that points to a database or other data store. Data sources define where data is stored for populating reports and views. There are several types of data sources, based on the type of connection or location of the data. For OLAP processing, you can use JDBC, JNDI, and bean data sources.

For a complete introduction to data sources, including defining a data source in JasperReports Server, see the JasperReports Server Administrator Guide.

## 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 the measures are the quantitative information (such as store sales and sales count).

For a detailed description of this process, see the Jaspersoft OLAP User Guide.

## 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 [XML/A Definitions](#).

For a detailed description of this process, see the Jaspersoft OLAP User Guide, which is available from Jaspersoft's community website <http://community.jaspersoft.com>.

## Defining an MDX Query

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

MDX is a query language used to define the contents of an OLAP view. It is also used when by Ad Hoc views built on OLAP client connections. 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].[2012].[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].[2012] 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 are not 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.

## Configuring XML/A

You can expose your Mondrian OLAP Connections to XML/A web services clients by configuring Mondrian XML/A Definitions. Ad Hoc and OLAP views can both 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 Microsoft Excel Pivot Tables. If you use the Community Project of Jaspersoft OLAP,

contact JasperSoft for more information. If you use a commercial edition, you can download this application from the Support Portal.

For more information on XML/A configuration, refer to [XML/A Definitions](#).

# Analyzing Data in a View

---

This chapter is meant to illustrate at a high level how to use an OLAP view to analyze your data. It has these sections:

- [OLAP Views](#)
- [Overview of OLAP View Analysis](#)
- [Using the OLAP View Tools](#)

The following sections assume you have installed the sample data.



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

## 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 to 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.

### To open the sample OLAP view

1. Click View > Repository to display the Repository panel.
2. In the Folders panel, expand the folder Organization > Analysis Components > Analysis Views.

A list of OLAP views appears in the Repository panel.

3. Click the name Foodmart Sample Analysis View to open the view.

The view displays the tool bar and navigation table.

For a detailed description of this process, see the [Jaspersoft OLAP User Guide](#).

# Overview of OLAP View Analysis

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, 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, which 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 2012 for stores in California?"

## Displaying Product Sales by Quarter

Jaspersoft OLAP is well suited to comparing numeric data across a period of time or by product category. That is how to answer the example question about snack food sales.

To open the sample OLAP view

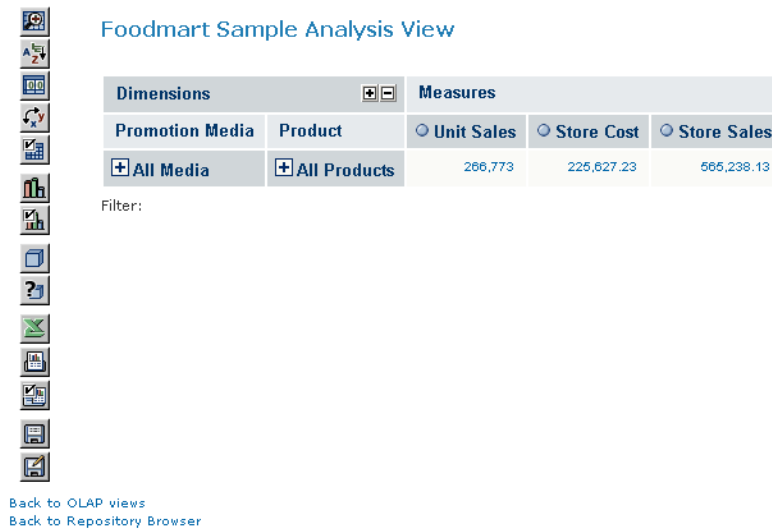
1. Click View > Repository to display the Repository panel.

- In the Folders panel, expand the folder Organization > Analysis Components > Analysis Views.

A list of OLAP views appears in the Repository panel.

- Double-click to open Foodmart Sample Analysis View.

The default navigation table appears along with the Jaspersoft OLAP tool bar.



The screenshot shows the 'Foodmart Sample Analysis View' interface. On the left is a vertical toolbar with various icons for OLAP operations. Below the toolbar are two links: 'Back to OLAP views' and 'Back to Repository Browser'. The main area contains a navigation table with the following structure:

Dimensions		Measures		
Promotion Media	Product	Unit Sales	Store Cost	Store Sales
+ All Media	+ All Products	266,773	225,627.23	666,238.13

Below the table is a 'Filter:' label.

**Figure 6: FoodMart Sample OLAP View**

To find the quarterly sales dollar amount in 2012 for stores in California

- Click the Change Data Cube tool bar button: .

The Change Data Cube dialog appears.

**Foodmart Sample Analysis View**

**Columns**

Measures

**Rows**

Promotion Media

Product

**Filter**

Customers

Education Level

Gender

Marital Status

Promotions

Store

Store Size in SQFT

Store Type

Time


Yearly Income

Dimensions		Measures		
Promotion Media	Product	Unit Sales	Store Cost	Store Sales
+ All Media	+ All Products	266,773	225,627.23	566,238.13

Filter:

*Figure 7: Change Data Cube*

The Change Data Cube dialog includes the following sections:

- Columns
  - Rows
  - Filters
2. Next to the Time filter, click the Move to Columns icon  to make Time a column.

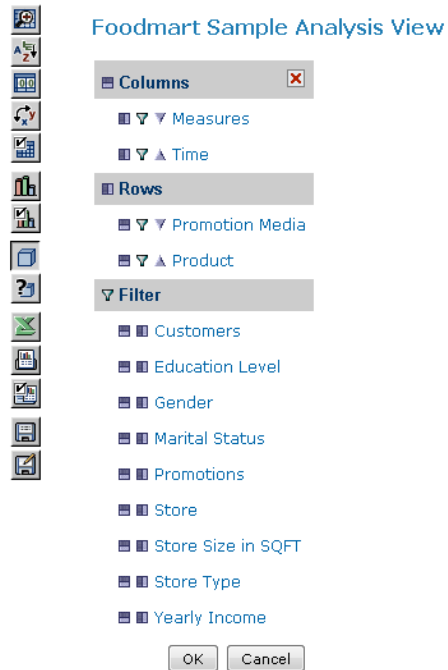


Figure 8: Adding a Time Column

3. In the Columns section, click Time to open a tree displaying the dimension members.
4. Expand Time by clicking **+** next to it, and select 2012.

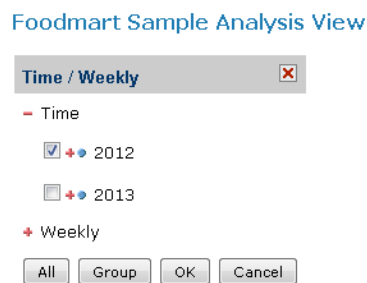


Figure 9: Defining the Time Filter

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



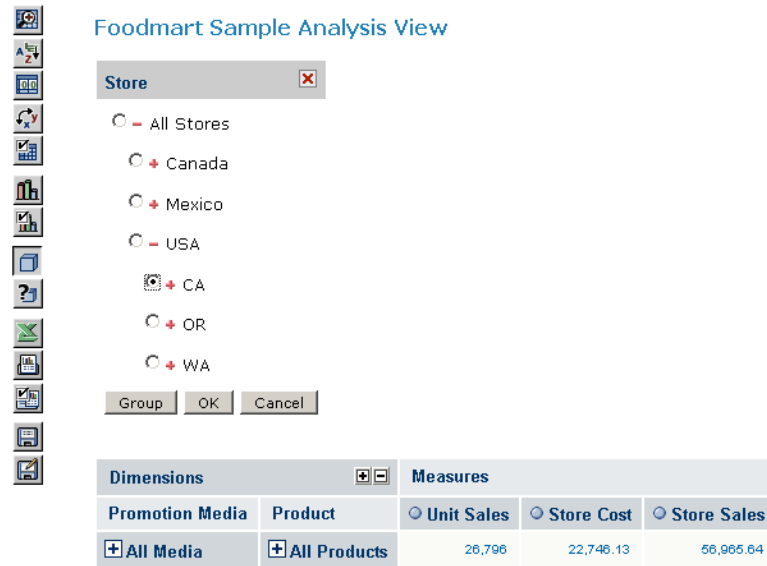
Changes you make in the Change Data Cube dialog are not applied to the view until you click OK again to close the dialog. Thus, while the Change Data Cube dialog is open, the view will not match your selections in the dialog.

6. In the Change Data Cube dialog, click Store in the Filter list to change the filter.



The Store filter appears.

- Expand All Stores by clicking **+** next to it, then expand USA, and select CA.



*Figure 10: Defining the Store Filter*

- Click OK to close the tree and return to the Change Data Cube dialog.
- Click Measures and clear the checkboxes next to Unit Sales and Store Cost to remove them.
- Click OK to accept the selections and close the Measures list, then click OK again to close the dialog.



Jaspersoft OLAP updates the view to match your selections. It is now filtered to show only the sales data for stores in California.


Dimensions		Measures
		Store Sales
		Time
Promotion Media	Product	<input type="radio"/> 2012
<input checked="" type="checkbox"/> All Media	<input checked="" type="checkbox"/> All Products	159,167.84

Filter: Store State=[Store].[USA].[CA]

Figure 11: 2012 Store Sales Volume for Snacks at California Stores


11. Ensure that the 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 is not 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 2012 to zoom into 2012 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 checkbox to clear it, and click OK.

The following navigation table appears:

Dimensions		Measures				
		Store Sales				
		Time				
Promotion Media	Product	<input type="radio"/> 2012	<input type="radio"/> Q1	<input type="radio"/> Q2	<input type="radio"/> Q3	<input type="radio"/> Q4
All Media	Snack Foods	19,072.42	4,193.45	4,646.08	4,808.16	5,424.73

Filter: Store State=[Store].[USA].[CA]

Figure 12: 2012 Quarterly Store Sales Volume for Snacks at California Stores

This view now shows the total sales for snack food in stores in California for 2012, broken down by quarter.






The following sections build on this example.

## Displaying Product Sales by Time Across Store Locations

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

In the following example, we'll use the Foodmart Sample Analysis View.

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

1. Click View > Repository to display the Repository panel.
  2. In the Folders panel, expand the folder Organization > Analysis Components > Analysis Views.  
A list of OLAP views appears in the Repository panel
  3. Click Foodmart Sample Analysis View to open it.
  4. Click .
- For details about using cube dimensions, see [Columns, Rows, and Filters](#).
5. Click the Move to Rows icon  next to the Store filter to create a row, then expand the Store row and select All Stores > USA. Click OK.
  6. Click the filter icon  next to Product in the Rows section, then click it and expand to and select All Products > Food > Snack Foods.
  7. Click OK twice to accept the selections.
  8. Click Zoom on Drill to activate it. When active, its tool bar button looks like this .
  9. Click the Edit Display Options tool bar button , click the Show all parent Columns check box to clear it, and click OK.
  10. In the table, click USA to zoom in.

The following navigation table appears.

Dimensions		Measures			
		Store Sales			
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	7,999.99	8,198.17

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

**Figure 13: 2012 Quarterly Snack Food Sales, Dollar Amount, for West Coast States**

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

## Charts and Exporting

When your view has the layout and data you need, you can use it to create a chart, or export it to a number of useful formats, such as PDF and Excel. For more information, see the Jaspersoft OLAP User Guide, available from Jaspersoft's community website <http://community.jaspersoft.com>.

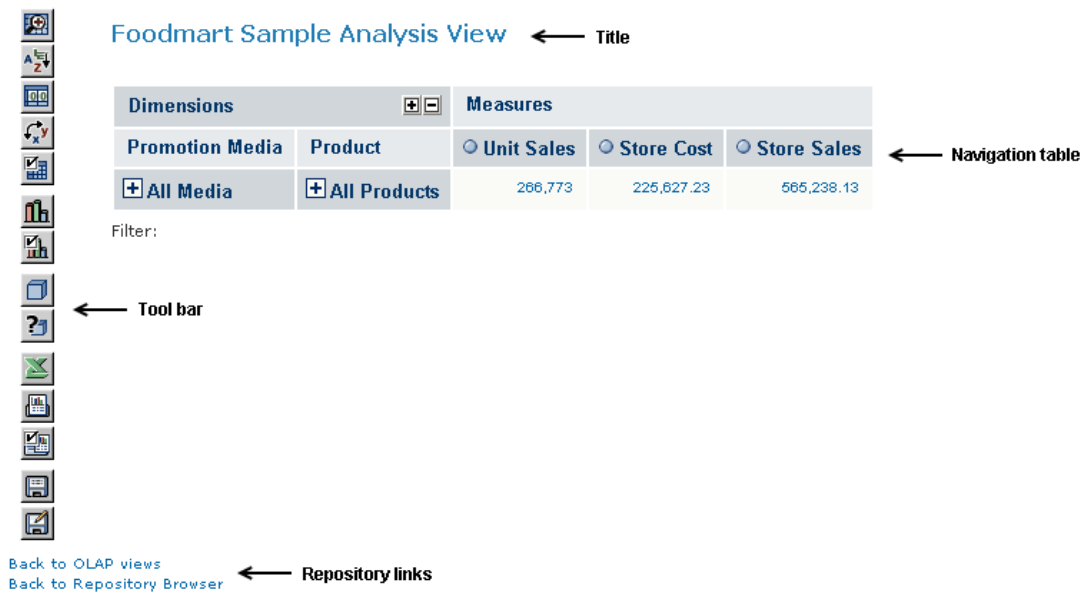
## Using the OLAP View Tools

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.

### Analysis Tool Bar and Navigation Table

In the tool bar, 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 14: A Sample OLAP View**

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 and average of the fact values, or more sophisticated calculated values such as year over year ratios and linear regressions.

**Typical Navigation Table** 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.

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]

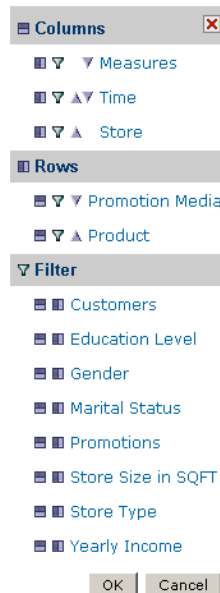
*Figure 15: Typical Navigation Table*

## 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.

## Columns, Rows, and Filters

Click  to open the Change Data Cube dialog.



*Figure 16: Change Data Cube Tools*

The Change Data Cube dialog configures the information that appears in the navigation table. In the table, the 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.

Dimensions					Measures						
					Store Sales						
					Time						
					2012	Q1			Q2		
Store	Store Country	Store State	Store City	Store Name	1	2	3	4	5	6	
All Stores	USA	WA			263,793.22	19,499.70	21,728.80	22,054.36	20,926.37	19,884.51	21,885.76
		CA			159,167.84	11,737.29	12,654.65	11,783.26	13,605.89	11,787.43	13,003.43
		OR			142,277.07	14,302.70	9,675.34	16,192.25	8,345.99	12,784.35	10,642.54
		OR	Salem		87,218.28	9,883.50	4,962.84	11,152.75	4,535.16	7,326.09	5,563.65
			Salem	Store 13	87,218.28	9,883.50	4,962.84	11,152.75	4,535.16	7,326.09	5,563.65
		WA	Tacoma		74,843.96	6,083.25	5,760.48	6,022.02	5,376.22	5,054.03	6,630.91
			Tacoma	Store 17	74,843.96	6,083.25	5,760.48	6,022.02	5,376.22	5,054.03	6,630.91
		OR	Portland		55,058.79	4,419.20	4,712.50	5,039.50	3,810.83	5,458.26	5,078.89
			Portland	Store 11	55,058.79	4,419.20	4,712.50	5,039.50	3,810.83	5,458.26	5,078.89
		CA	Los Angeles		54,545.28	4,167.72	5,570.17	3,999.08	3,917.49	3,396.00	4,370.95
			Los Angeles	Store 7	54,545.28	4,167.72	5,570.17	3,999.08	3,917.49	3,396.00	4,370.95
			San Diego		54,431.14	4,722.87	3,854.89	4,720.27	4,296.07	4,345.29	4,335.92
			San Diego	Store 24	54,431.14	4,722.87	3,854.89	4,720.27	4,296.07	4,345.29	4,335.92
		WA	Bremerton		52,896.30	3,437.91	4,530.33	4,517.53	4,235.60	3,700.04	5,080.67
			Bremerton	Store 3	52,896.30	3,437.91	4,530.33	4,517.53	4,235.60	3,700.04	5,080.67
			Seattle		52,644.07	4,037.00	4,154.51	4,569.13	4,444.06	4,557.58	3,531.22
			Seattle	Store 15	52,644.07	4,037.00	4,154.51	4,569.13	4,444.06	4,557.58	3,531.22
			Spokane		49,634.46	3,244.88	4,606.84	3,778.39	4,109.22	3,809.01	3,717.19
			Spokane	Store 16	49,634.46	3,244.88	4,606.84	3,778.39	4,109.22	3,809.01	3,717.19
		CA	Beverly Hills		45,750.24	2,629.09	2,836.37	2,738.43	5,065.00	3,867.80	3,864.35

**Figure 17: 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.



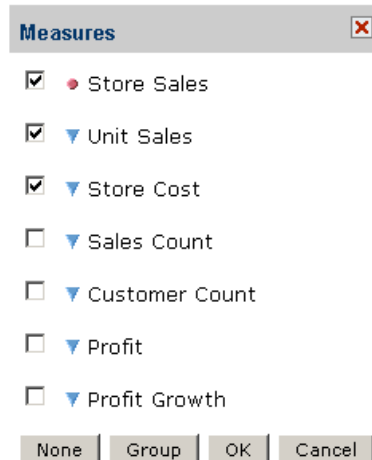


**Figure 18: Measures Dialog**

2. Click a checkbox to select its measure.

To move a measure within its section of the dialog



1. Click  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 19: Measures Dialog with Moving Enabled**


2. Click any triangle. The measure selected in [Click next to the measure. The icon changes color to red](#) . , Store Sales, moves down one position in the list.
3. Click a checkbox 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.

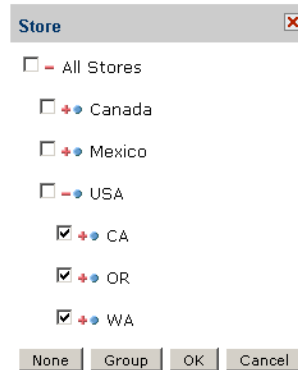
To move a dimension into the Columns or Rows section of the Change Data Cube dialog

- 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 20: STORE Dimension Collapsed**

The root level of the STORE dimension contains a special member called ALL STORES. Clicking the checkbox 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 21: STORE Dimension Expanded**

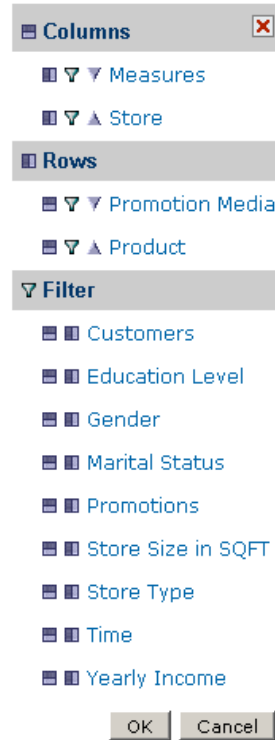
## Filter

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 22: TIME Dimension as a Filter**

2. Click the dimension.

The root level of the dimension appears. In this example, look at the TIME dimension.




**Figure 23: 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.

## 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 24: MDX Query Editor**

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

- A SELECT statement determines the data sets that 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 this example, [TIME].[2012] 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 <https://learn.microsoft.com/en-us/analysis-services/multidimensional-models/mdx/mdx-query-the-basic-query?view=asallproducts-allversions>.



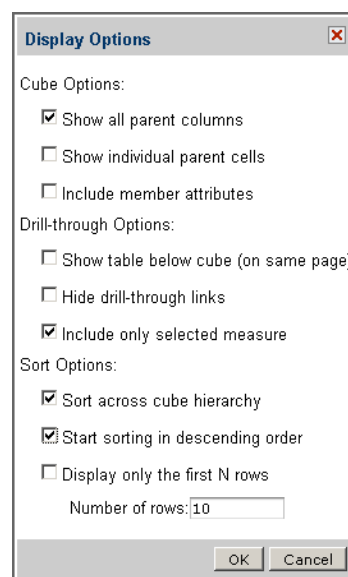
The rest of the examples in this chapter start with the query in [MDX Query Editor](#) above. To recreate the examples yourself, start with that query.

## Sort Options

The sort options include the following:

- Sort into a 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 25: 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 hierarchically, within their parent member. Take, for instance, a crosstab showing the store sales for all stores in the United States. The dimension hierarchy might be as follows:

- All Stores
- USA
  - CA
  - Alameda
  - Beverly Hills

OR

- Portland
- Salem
- WA
- Bellingham
- Seattle

All Stores is above USA in the hierarchy, which is above CA in the hierarchy, and so on.


In the above list, the STORE COUNTRY (USA) and STORE STATE (CA, OR, WA) are hierarchy boundaries. STORE CITY items (such as Alameda and Portland) are hierarchy members.

You can use the Sort across cube hierarchy option to change the navigation table so the member measures are sorted without regard to the hierarchies that contain them. For instance, you may want to list all of your USA stores based on their sales, highest to lowest, regardless of the states they are in, as illustrated below.

Dimensions				Measures	
Store	Store Country	Store State	Store City	Store Sales	Unit Sales
All Stores	USA	OR	+ Salem	87,218.28	41,580
		WA	+ Tacoma	74,843.96	35,257
		OR	+ Portland	55,058.79	26,079
		CA	+ Los Angeles	54,545.28	25,663
			+ San Diego	54,431.14	25,635
		WA	+ Bremerton	52,896.30	24,576
			+ Seattle	52,644.07	25,011
			+ Spokane	49,634.46	23,591
		CA	+ Beverly Hills	45,750.24	21,333
		WA	+ Yakima	24,329.23	11,491
			+ Bellingham	4,739.23	2,237
			+ Walla Walla	4,705.97	2,203
		CA	+ San Francisco	4,441.18	2,117
			+ Alameda		

Filter: Year=[Time].[2012]

**Figure 26: Sorting Across the Cube Hierarchy**

To do this, set up your analysis view to show the dimensions you want to display in rows, and the measures you want to display in columns. Click the Sort icon  next to the measure that you want to sort on. Then open the Display Options dialog as described above, and click Sort 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 checkbox.

## 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. You 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 checkbox to disable the row limitation.



## 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

### Show all parent columns

The Show all parent columns checkbox displays a header for every dimension column. When the checkbox 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 [Comparing Layout: All Parent Columns and No Parent Columns](#), the All parent column 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 that all parent columns is often used with Zoom on Drill to provide complete context after zooming in.

Dimensions <input type="checkbox"/> <input type="checkbox"/>			Measures	
Store	Store Country	Store State	○ 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
		[+] OR	142,277.07	67,659
		[+] WA	263,793.22	124,366

Filter: Year=[Time].[2012]

← All parent columns

No parent columns ↓

Dimensions <input type="checkbox"/> <input type="checkbox"/>	Measures	
Store	○ Store Sales	○ Unit Sales
[-] All Stores	565,238.13	266,773
[-] USA	565,238.13	266,773
[+] CA	159,167.84	74,748
[+] OR	142,277.07	67,659
[+] WA	263,793.22	124,366

Filter: Year=[Time].[2012]

**Figure 27: Comparing Layout: All Parent Columns and No Parent Columns**

## Show individual parent cells


The Show individual parent cells checkbox determines whether redundant dimension hierarchy member labels are displayed or hidden. The checkbox takes effect when the Show all parent columns checkbox is checked. The following figure shows the example in [Comparing Layout: All Parent Columns and No Parent Columns](#) 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 can result.

Dimensions 			Measures	
Store	Store Country	Store State	Store Sales	Unit Sales
 All Stores			565,238.13	266,773
All Stores	 USA		565,238.13	266,773
All Stores	USA	 CA	159,167.84	74,748
All Stores	USA	 OR	142,277.07	67,859
All Stores	USA	 WA	263,793.22	124,366


Filter: Year=[Time].[2012]


**Figure 28: Showing Individual Parent Cells**

## 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 are measured display, even when they have no value.

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

Dimensions			Measures	
Store	Store Country	Store State	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
		[+] OR	142,277.07	67,659
		[+] WA	263,793.22	124,366

Filter: Year=[Time].[2012]

Five dimensions rows and two measure columns

Dimensions	Store				
	[-] All Stores		All Stores		
		[-] USA	USA	[+] CA	[+] OR
Measure					
Store Sales	565,238.13	565,238.13	159,167.84	142,277.07	263,793.22
Unit Sales	266,773	266,773	74,748	67,659	124,366

Filter: Year=[Time].[2012]

Two measure rows and five dimension columns

**Figure 29: Swapping Axes**

## 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.

## 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.

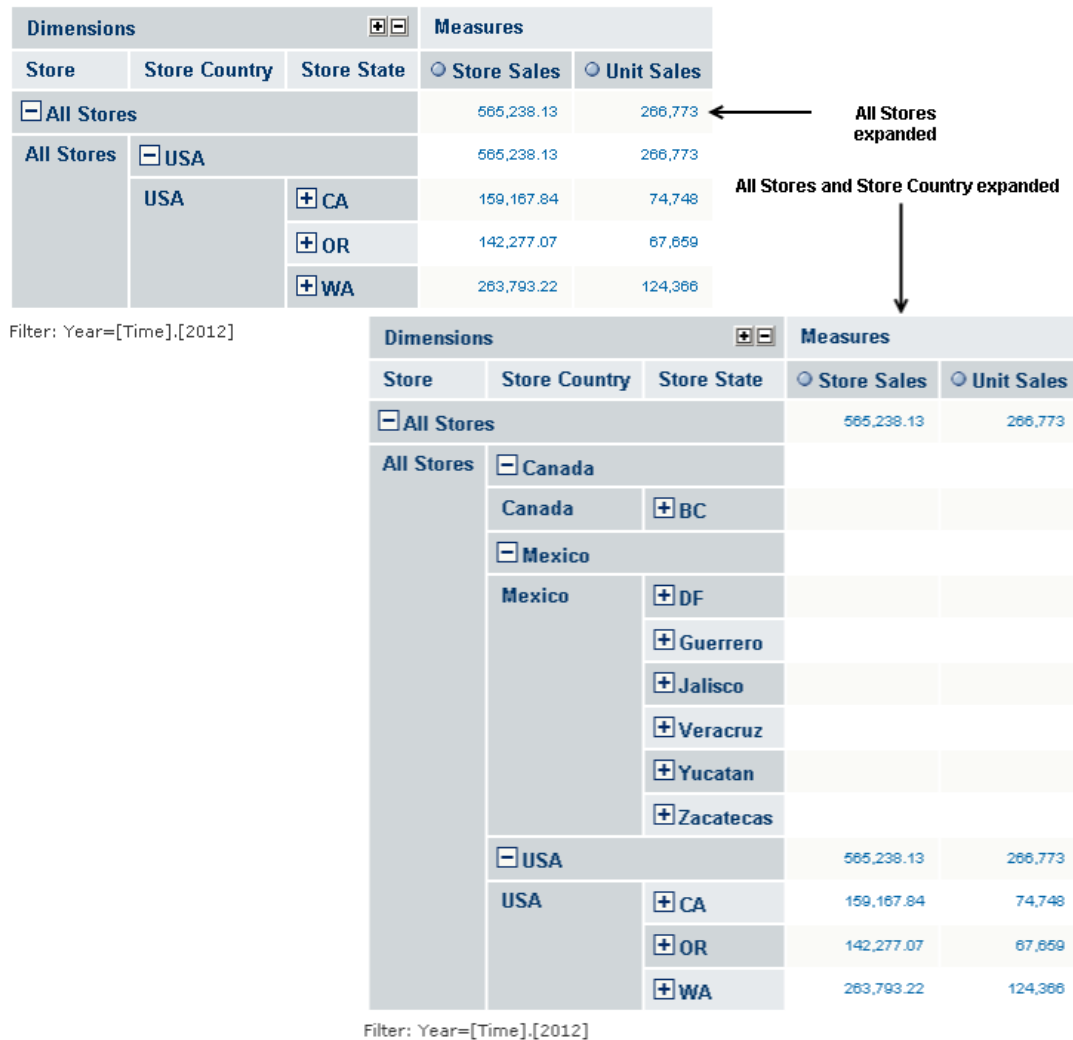




Figure 30: Expanding Positions

## 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.

## Drill-through

Drill-through displays the detailed data that produced a measure's value, if that the value is an aggregation of 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 is 24,450 rows.

To disable drill-through, open the Display Options dialog ([Sort Options in Display Options Dialog](#)) 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	2012	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2012	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2012	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2012	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2012	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2012	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2012	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2012	Q1	1	2	1	Drink	Beverages
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2012	Q1	1	2	1	Drink	Dairy
CA	Beverly Hills	Store 6	23688	Gourmet Supermarket	2012	Q1	1	2	1	Food	Baked Goods

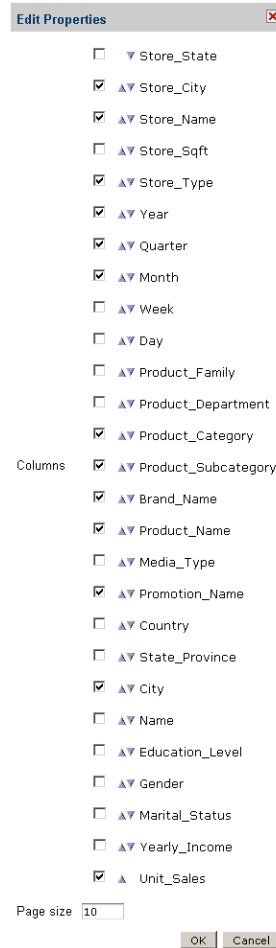
Page 1/2,445 ▶▶ Goto Page  Rows/page

**Figure 31: 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 icon 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.

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 32: Edit Properties Dialog for a Drill-through Table**

The Edit Properties dialog has three controls:

- Measures. Though not labeled in this dialog, the items in the list above the Columns label are measures. To include a measure in the table, select its checkbox. To hide a measure from view, clear its checkbox.
- Columns. To include a column in the table, select its checkbox. Use the up-down triangles ▲▼ to move the column to the left or right. To hide a column from view, clear its checkbox.
- Page size. Specifies the number of rows on a page.



# 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 are 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 Jaspersoft OLAP User Guide. 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 Attributes](#)
- [Configuring CZS's Ad Hoc View](#)
- [Reference Material](#)



This chapter assumes that you have the administrator role. It describes a number of tasks that only administrators can perform.

# 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 is using 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 Ad Hoc view to track quarterly 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 Ad Hoc 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 attributes.

## Overview of CZS's Process

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

Step	Described in Section ...
1	Identified the dimensions on which to base access control. CZS chose the Geographical Area and Product

Step		Described in Section ...
	Department dimensions.	
2	Identified and created access roles. CZS identified two roles: one for managers and another for sales reps. Both are granted access to the Ad Hoc view.	<a href="#">Determining Roles</a>
3	Assigned the appropriate roles to each user based on each employee's responsibilities.	<a href="#">Selecting Users for the Roles</a>
4	Identified the attributes that are defined for each user. CZS identified five attributes: Country, Region, State, Cities, and ProductDepartment.	<a href="#">Attributes and Variable Substitution</a> and <a href="#">Assigning Attributes</a>
5	Defined the correct values for each user's attributes by editing user accounts.	<a href="#">Defining Attributes for Users</a> and the <a href="#">JasperReports Server Administrator Guide</a>
6	Created an AGXML (access grant definition XML) file that defines the access granted to users with each role and attribute.	<a href="#">Using Attributes in an Access Grant Definition</a> and <a href="#">Reference Material</a>
7	Created a Mondrian connection that pointed to their sales data and included the sales access grant definition.	<a href="#">Creating a Mondrian Connection</a>
8	Created the Ad Hoc view that points to the Mondrian connection.	<a href="#">Creating a Sales Numbers Ad Hoc View</a>
9	Tested the views of various users.	<a href="#">Testing the Results</a>

# Understanding Access Grant Definitions and 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 [Access Grant Definition for CZS](#).

## About 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="[Geographical 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]">
          <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].[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>
</Roles>
```

```

</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].[Irrelevances]" 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. To avoid this complexity, CZS defined user-level 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.

## Attributes and Variable Substitution

An attribute is a name-value pair defined at the user, role, organization, or server instance level; in this example, they correspond 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 Ad Hoc 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 attributes, this would only be possible if CZS's access roles were defined along geographical lines. Instead, CZS defines attributes that indicate each user's geographical affiliations: Rita is responsible for California and Alexi is responsible for Osaka.



For the sake of simplicity, this example only includes user-level attributes. Despite this, variable substitution can be used in conjunction with attributes defined at any level.

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 for the specific user.

CZS also defined 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 attributes, is shown in [Access Grant Definition for CZS](#).

## Configuring CZS's Ad Hoc View

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

## Defining a Sales Numbers Cube

The following sections describe the measures and dimensions in the cube that represent CZS's sales data. These data are displayed to CZS's users in an Ad Hoc view that is built from the measures and dimensions in the cube.

## 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_2012` fact table. This fact table includes three measures: unit sales, store cost, and store sales.

## Dimensions

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

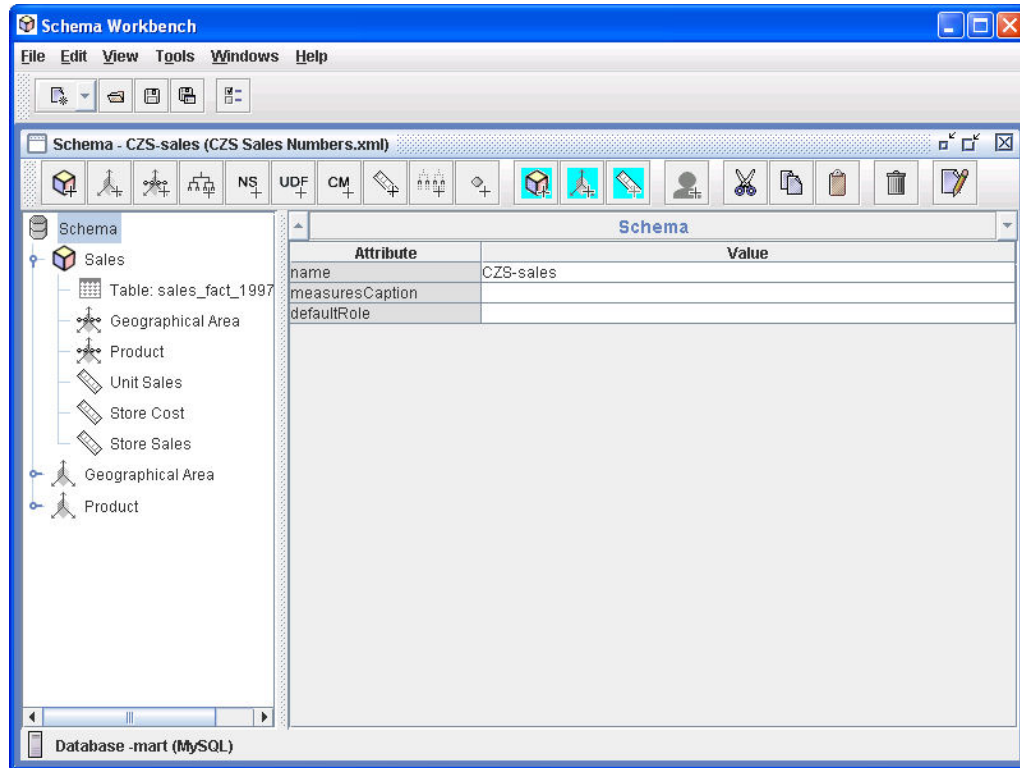
- The geographical dimension includes these levels: country, region, state, and city.
- The product dimension describes the merchandise being sold and includes these levels: product family, product line, and product.

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

## 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 33: 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 [OLAP Schema](#).

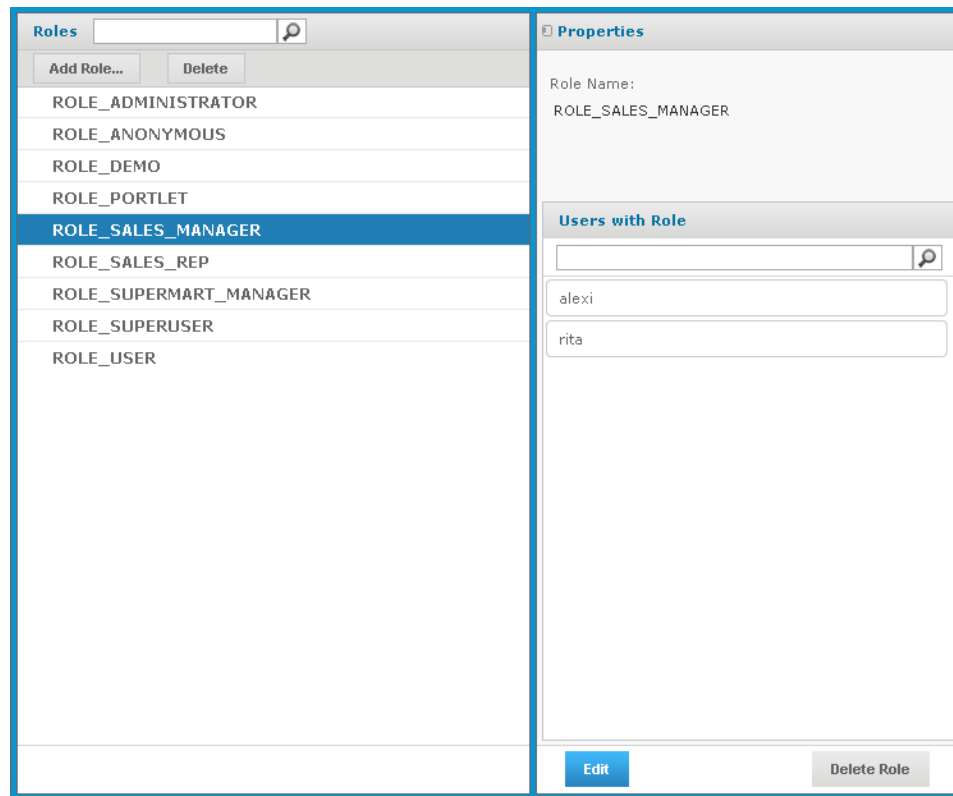
## Determining Roles

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 Ad Hoc view. For details about creating roles and assigning privileges, refer to the boot JasperReports Server Administrator Guide.

The following shows CZS's ROLE\_SALES\_MANAGER:



**Figure 34: CZS's ROLE\_SALES\_MANAGER User Role**

## Selecting Users for the Roles

CZS defined a user account for each of their employees. It then assigned roles to each account; the roles are consistent with the employees' levels of responsibility.

Rita	ROLE_SALES_MANAGER
Pete	ROLE_SALES_REP
Yasmin	ROLE_SALES_REP
Alexi	ROLE_SALES_MANAGER

For details about creating users, refer to the JasperReports Server Administrator Guide.

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

**Properties**

User name:  
**Rita**

User ID:  
rita

Email:

User is enabled.

Roles Assigned:  
ROLE\_SALES\_MANAGER,ROLE\_USER

Profile Attributes:

Attribute name	Attribute value
Country	USA
State	CA
Cities	Sacramento, San Francisco, Los Angeles
Region	West
ProductDepartment	Televisions, Wireless Devices

[Edit](#) [Login as User](#) [Delete User](#)

**Figure 35: Rita's User Account**

## Assigning Attributes



Attributes can be defined for each server, organization, role, or user. See the JasperReports Server Administrator Guide for more information about defining the attributes.

Notice the attributes listed below the user's roles in [Rita's User Account](#). CZS implemented them to take 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 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 [Attributes and Variable Substitution](#) refers to these attributes.

- One describes the products that 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 these 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 that a subset of the data Rita sees.

CZS assigned the following attributes to their users:

Attributes of CZS Users					
User	User 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 defining and leveraging attributes:

1. Defined the required attributes for each user by editing the associated account. These attributes are referred to by variable substitution in the AGXML file.
2. Created an access grant definition (AGXML file) that refers to these attributes.
3. Used the access grant definition in a Mondrian connection.
4. Used the Mondrian connection in the Ad Hoc Editor.

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

- [Defining Attributes for Users](#)
- [Using Attributes in an Access Grant Definition](#)
- [Creating a Mondrian Connection](#)
- [Creating a Sales Numbers Ad Hoc View](#)

## Defining Attributes for Users

Attributes can be created for each user, role, or organization defined in the server. At the user level, they are maintained as part of the account, which you can edit by clicking Manage > Users. CZS edited each of their users to include the attributes they mapped out during their planning phase. [Figure 36](#) shows Rita's attributes in the Manage > Users page.

The screenshot displays the 'Users' management page. On the left, a table lists users, with 'rita' selected. On the right, the 'Properties' panel is open to the 'Attributes' tab, showing a list of attributes for Rita's user account.

Users		
Username	Full Name	Organization
CaliforniaUser	California User	organization_1
alexi	Alexi	organization_1
demo	Demo User	organization_1
jasperadmin	jasperadmin User	organization_1
joesuser	Joe User	organization_1
pete	Pete	organization_1
<b>rita</b>	<b>Rita</b>	<b>organization_1</b>
yasmin	Yasmin	organization_1

Properties		
Attributes		
Attribute name	Attribute value	
Country	USA	Remove
State	CA	Remove
Cities	Sacramento,San Francisco,Los	Remove
Region	West	Remove
ProductDepartment	Televisions,Wireless Devices	Remove
		Add

Buttons: Save, Cancel

**Figure 36: Editing Rita's User Attributes**

For more information, refer to the JasperReports Server Administrator Guide.

## Using Attributes in an Access Grant Definition

With the attributes defined, CZS took advantage of them using variable substitution in their access grant definitions, as described in [Attributes and Variable Substitution](#).

CZS's access grant definition (cza-access-grant.agxml) is shown in [Access Grant Definition for CZS](#). It utilizes the attributes discussed in this section (Country, Region, State, Cities, and ProductDepartment).

## Creating a Mondrian Connection

CZS created a Mondrian connection that included the cza-sales-grant.agxml access grant definition file. Ad Hoc views, reports, OLAP views, and XML/A connections that use this Mondrian connection impose the data level security rules defined in the AGXML file.

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

## Creating a Sales Numbers Ad Hoc View

Having created the attributes, access grant definition, and a Mondrian connection, CZS then created the actual Sales Numbers Ad Hoc view to run against the Mondrian connection. This Ad Hoc view pointed to the Mondrian connection that includes the cza-sales-grant.agxml access grant definition file. CZS used the Ad Hoc Editor to add members of the geographical dimension and product dimensions, as well as the store sales, unit sales, and store cost measures into a crosstab.

Once the Ad Hoc 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 Ad Hoc views.

For instructions on access control, refer to the JasperReports Server Administrator Guide. For instructions on creating OLAP views and Ad Hoc views, refer to the OLAP views Jaspersoft OLAP User Guide.

## Testing the Results

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

To test the access granted to users on data in the CZS Sales Numbers Ad Hoc 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. In the Search field, enter CZS and press return
5. Click CZS Sales Numbers Ad Hoc View.

The view appears.

6. Test the view to ensure that it only displays data the user should see and does not hide data the user should see.
7. Click Log Out to return to your Home page, then login as each of the other users.

When reviewing the CZS Ad Hoc views below ([Rita's View](#) to [Alexi's View](#)), note the different access each user has to the data.

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).

**CZS Ad Hoc View (Mondrian Connection)**

Columns: Store Sales, Unit Sales, Store Cost

Rows: State, City, Product Family, Product Department

**CZS Sales Numbers Ad Hoc View**

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

**Figure 37: Rita's View**

Pete can only see television data about San Francisco:

**CZS Sales Number Ad Hoc View (OLAP)**

Columns: Store Sales, Unit Sales, Store Cost

Rows: City, Product Department

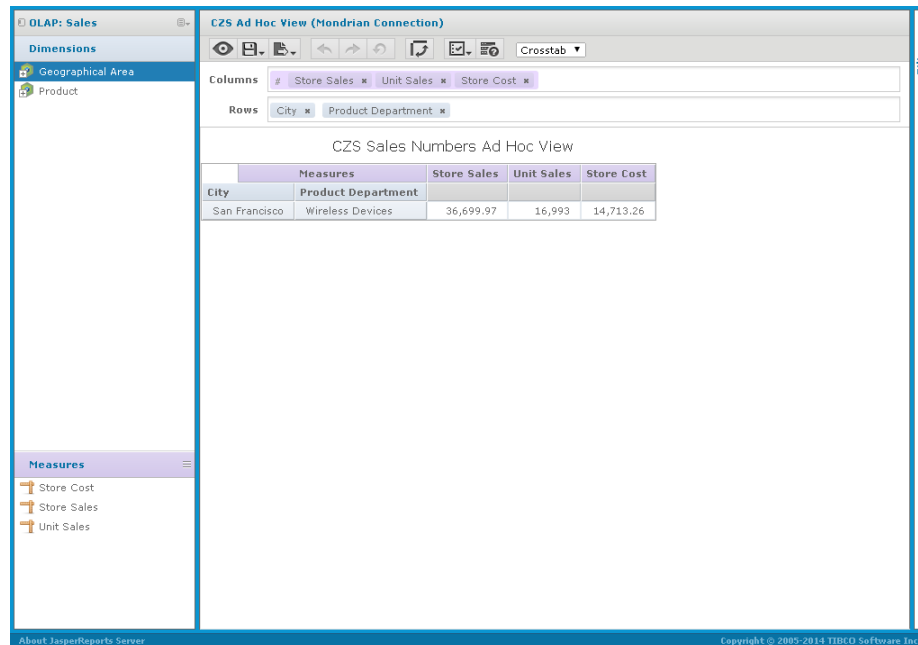
**CZS Sales Numbers Ad Hoc View**

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

**Figure 38: Pete's View**

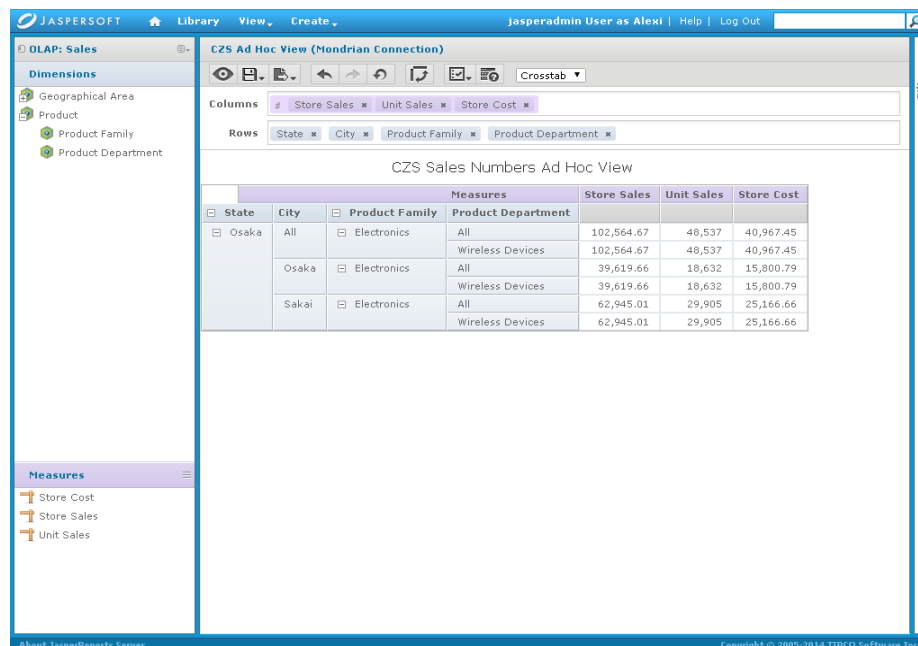
Yasmin can only see wireless devices data about San Francisco.





**Figure 39: Yasmin's View**

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



**Figure 40: Alexi's View**

# Reference Material

## OLAP Schema

The CZS-sales.xml OLAP schema defines a cube that is based on the sales\_fact\_2012 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" />
      </Join>
    </Hierarchy>
  </Dimension>
</Schema>
```

```

<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>

```

## Access Grant Definition for CZS

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 attributes that determine the data the user can see in the Ad Hoc 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>

<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>

```

# 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](#)

## Understanding the Design Concepts

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

## Introduction to Dimensional Modeling

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 Kris were married on June 15 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 would 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.

## 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 do not 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 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 an 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 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 those listed in the following table.

#### **Creating a Dimensional Model**

<b>Process in Dimensional Modeling</b>	<b>Examples of Process Applied to Specific Data</b>
Phrasing the business problems to be	<ul style="list-style-type: none"> <li>• How does customer satisfaction affect</li> </ul>

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



Process in Dimensional Modeling	Examples of Process Applied to Specific Data
	<p>normal range?</p> <ul style="list-style-type: none"> <li>Why are the Satisfaction Scores and Number of Repeat Transactions consistently lower for one particular store than the average across the business?</li> </ul>

## Applying the Model

By following these processes, you define a model that describes your data in terms of dimensional analysis.

## 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. Our goal in this section is to answer this question: what type of alcoholic beverage generates the most revenue?

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. 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	58,965.64


Filter: Month=[Time].[2012].[Q4].[12]

**Figure 41: Top-level Summary for December**

```
select {[Measures].[Store Sales]} ON COLUMNS,
Hierarchize({[Product].[All Products]}) ON ROWS
```

```
from [Sales]
```

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


To drill-down, click  next to the ALL PRODUCTS member. The next level of the PRODUCT hierarchy, Product Family, 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].[2012].[Q4].[12]

**Figure 42: Expand All Products**

```
select {[Measures].[Store Sales]} ON COLUMNS,
Hierarchize(Union({[Product].[All Products]}, [Product].[All
Products].Children)) ON ROWS
from [Sales]
where [Time].[2012].[Q4].[12]
```

3. Drill down again, this time by zooming on the Drink product family. Use  on DRINK.

Dimensions			Measures
Product	Product Family	Product Department	Store Sales
All Products	Drink	 Alcoholic Beverages	1,441.48
		 Beverages	2,855.45
		 Dairy	705.10

Filter: Month=[Time].[2012].[Q4].[12]

**Figure 43: Zoom on the Drink Product Family**

```
select {[Measures].[Store Sales]} ON COLUMNS,
[Product].[Drink].Children ON ROWS
from [Sales]
where [Time].[2012].[Q4].[12]
```

4. By zooming, we could drill down to the lowest level of data in the cube. Let us 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].[2012].[Q4].[12]

**Figure 44: 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].[2012].[Q4].[12]
```

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



5. Let us see if that finding holds true across different stores. To do so, we add another dimension, which creates a crossjoin. Analytic views make creating a crossjoin easy.
  - a. Select , make STORE a row, and select All Stores> USA > CA (for instructions on adding a dimension, refer to [Cube Configuration](#)). 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].[2012].[Q4].[12]

**Figure 45: 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].[2012].[Q4].[12]
```

- b. Deselect the , then use  to drill down to navigate to intersections of the store and city data. For example, under both the BEER and WINE subcategories, drill-down on ALL STORES, then USA, then CALIFORNIA to examine the data on

the California stores. Notice that you are 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									Measures	
Product	Product Family	Product Department	Product Category	Product Subcategory	Store	Store Country	Store State	Store City	Store Sales	
All Products	Drink	Alcoholic Beverages	Beer and Wine	Beer	All Stores	USA	CA	Alameda		
								Beverly Hills	34.44	
								Los Angeles	44.88	
								San Diego	43.82	
								San Francisco		
					Wine	All Stores	USA	CA	Alameda	
					Beverly Hills				127.60	
					Los Angeles				140.62	
					San Diego				91.06	
					San Francisco				12.78	

Filter: Month=[Time].[2012].[Q4].[12]

**Figure 46: Stores in California**

```
select {[Measures].[Store Sales]} ON COLUMNS,
  Crossjoin([Product].[Drink].[Alcoholic Beverages].[Beer and Wine].Children,
  [Store].[USA].[CA].Children) ON ROWS
from [Sales]
where [Time].[2012].[Q4].[12]
```

With Jaspersoft OLAP, the data can be “sliced and diced” in many ways, allowing you to focus the viewer’s attention on the most relevant data. There are usually several ways to view the same data. The previous steps showed some of these ways. We could also make the STORE dimension a column. This might be a good approach when you have many different products to compare. Doing so results in a two-by-five grid, with the single-dimension layout becoming a two-dimension layout.

Dimensions					Measures				
Product	Product Family	Product Department	Product Category	Product Subcategory	Alameda	Beverly Hills	Los Angeles	San Diego	San Francisco
All Products	Drink	Alcoholic Beverages	Beer and Wine	Beer		34.44	44.88	43.82	
				Wine		127.60	140.62	91.06	12.78

Filter: Month=[Time].[2012].[Q4].[12]

**Figure 47: Stores as Columns Instead of Rows**

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

```
from [Sales]
where [Time].[2012].[Q4].[12]
```

For another example, let us make the table easier to understand by switching the rows and columns. This is called “pivoting”, that is, we are rotating this section of the cube. Pivoting gives us a five-by-two grid of data cells, which is often easier to read online and is better suited to printing on U.S. Letter-sized paper. While the previous example compares product subcategories by store, swapping the columns and rows lets us compare each store’s results by product.

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].[2012].[Q4].[12]

**Figure 48: 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].[2012].[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. Perhaps this points to supply chain problems.

## Leveraging the Ad Hoc Editor

The model we defined in [Designing the Model](#) and implemented in [Applying the Model](#) can also be leveraged by the Ad Hoc Editor. Select the sample FoodMart Mondrian connection as your data source when you open the editor. Then slice, dice, and drill in the same manner as you have done in Jaspersoft OLAP. The resulting Ad Hoc view can be saved as a report, which in turn can be scheduled for email distribution and incorporated into dashboards. [FoodMart Sales data in an Ad Hoc View](#) shows an Ad Hoc view of the same data as shown in the OLAP view in [Pivoting the View](#).

		Measures	Store Sales			
		(All)	All Products			
		Product Category	All	Beer and Wine		
		Product Subcategory	All	All	Beer	Wine
		Brand Name	All	All	All	All
		Product Name	All	All	All	All
(All)	Store State	Store City				
All Stores	All	All	56,965.64	1,441.48	361.92	1,079.56
	CA	All	16,839.01	495.10	123.14	371.96
		Alameda				
		Beverly Hills	5,481.64	162.04	34.44	127.60
		Los Angeles	5,211.38	185.40	44.88	140.52
		San Diego	5,684.35	134.88	43.82	91.06
		San Francisco	461.64	12.78		12.78

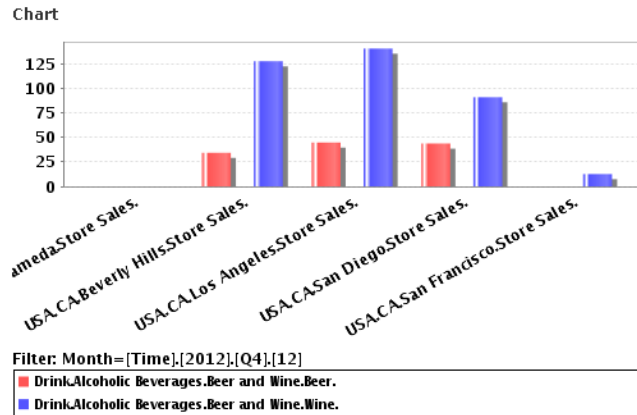
**Figure 49: FoodMart Sales data in an Ad Hoc View**

In the course of this example, we have unearthed questions we had not 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].[2012].[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 50: 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 at 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. You are looking at individual rows of data to understand this one store's wine sales in December of 2012 (table is truncated).

 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	Product_Department
CA	San Francisco	Store 14	22478	Small Grocery	2012	Q4	12	1	18	Drink	Alcoholic Beverages
CA	San Francisco	Store 14	22478	Small Grocery	2012	Q4	12	1	18	Drink	Alcoholic Beverages
CA	San Francisco	Store 14	22478	Small Grocery	2012	Q4	12	2	24	Drink	Alcoholic Beverages
CA	San Francisco	Store 14	22478	Small Grocery	2012	Q4	12	50	4	Drink	Alcoholic Beverages

**Figure 51: Drill-through Table of San Francisco Beer And Wine Sales**

## Maintaining the System

This section includes information about system maintenance, including:

- [XML/A Definitions](#)

- [Monitoring the System](#)
- [Maintaining Tables](#)
- [Clearing the Cache](#)

## 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 [https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2005/ms187178\(v=sql.90\)](https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2005/ms187178(v=sql.90)).

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 SQL Server Analysis Services and Mondrian OLAP server, and to various data sources, such as MySQL or PostgreSQL. However, take care when using XML/A, as not all providers behave the same way.

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 SQL Server Analysis Services. To purchase ODBO Connect, please contact your sales representative.

You can create an OLAP view against an XML/A connection just as you did against a Mondrian connection in [OLAP Views](#). The steps are nearly identical. Simply select an XML/A connection instead of a Mondrian connection when prompted for the connection. Ad Hoc views can also be built against XML/A connections.

For more information about creating XML/A sources and connections, refer to the Jaspersoft OLAP User Guide , which is available at the [Jaspersoft Community site](#).



## 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.propertiesfile. When debugging an exception, set the logging level to DEBUG for the specific classes mentioned in the exception.
- The log4j documentation is available at: <https://logging.apache.org/log4j/2.x/manual/index.html>. For JasperSoft-specific information on logging, see the JasperSoft OLAP User Guide and JasperReports Server Administrator Guide.

If you need to contact Technical Support, they may request that you use the log collector feature to supply your logs to them so they can understand what may be causing any issues you may encounter. This feature is also described in JasperReports Server Administrator Guide.



For information on contacting Technical Support, see [Troubleshooting Information for Technical Support](#).

## 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 tables 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 to 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 that updates or 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 than the others.

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 afterward 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 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.

## 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 a 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.

For information about creating XML/A sources and connections, see the *Jaspersoft OLAP User Guide*.

## Tuning Performance

### 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.

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 and Response**

<b>Step</b>		<b>Resources Relevant to Bottleneck Analysis</b>
1	MDX is generated	
2	MDX is passed to the local OLAP engine	CPU and memory of front-end negligible
	MDX is passed remotely using XML/A	Network latency between front-end and OLAP engine
3	MDX is processed by the 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
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 run 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 the many OLAP engines in a cluster. See [Load Balancing](#).

## 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. [Typical Performance Profile and MDX Report](#) shows two examples: the Profiling OLAP view and the MDX frequency report.

#### Profile View

Measures								
Duration in ms	Number of ops	SQL queries	MDX queries	Cache hits	Aggregate hits	Cache hit ratio	Agg hit ratio	Avg SQL queries per MDX query
712	15	0	13	0	0	NaN%	0%	0

Filter:

## MDX Query Frequency

freq	query	avg. ms
10	select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]} ON COLUMNS, Hierarchy([Store].[USA].Children) ON ROWS from [Sales]	52
3	select {[Measures].[Total Sale Amount], [Measures].[Number of Sales], [Measures].[Avg Sale Amount], [Measures].[Avg Time To Close (Days)], [Measures].[Avg Close Probability]} ON COLUMNS, NON EMPTY {[Account Categorization].[All Accounts], [Clo...]	59
1	select {[Measures].[Duration in ms], [Measures].[Number of ops], [Measures].[SQL queries], [Measures].[MDX queries], [Measures].[Cache hits], [Measures].[Aggregate hits], [Measures].[Cache hit ratio], [Measures].[Agg hit ratio], [Measures].[Avg S...]	8

**Figure 52: Typical Performance Profile and MDX Report**

Enabling performance profiling may have a minor impact on the overall performance of the system, since Jaspersoft OLAP must run additional queries 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 to performance profilings, 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 server 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 are 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 do not 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. Having a benchmark is an important part of effective performance tuning.

## Jaspersoft OLAP Tuning Process and Options

Building a high-performance database for the largest scale of data volume can involve various optimizations that 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. The following table summarizes hardware level performance tuning.

### Hardware Level

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 Issue	Comments
	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.

## RDBMS Level

The biggest decision at this level is which RDBMS to use. Oracle, MySQL, PostgreSQL, Microsoft SQL Server, 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 table summarizes RDBMS level performance tuning.

Optimization Issue	Comments
Temp space	The number of large joins required by multidimensional queries makes

Optimization Issue	Comments
	heavy use of temp space for storing intermediate results. A SQLException is generated 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.
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 do not 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	Using read-only connections for OLAP may allow for faster cursor access.

Optimization Issue	Comments
connections and redo/undo logs	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.

## 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 “sub-dimension” 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 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 look up 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 generates 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_2013` 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 matter, 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 the aggregate tables that 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 the 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 from [SourceForge](#). 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 have to use a trial-and-error approach to determine the best ones for your system.

## 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.

## Load Balancing

For systems with many simultaneous users, multiple application servers and load balancing 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 is installed similarly on each computer, but the metadata will be configured slightly differently.

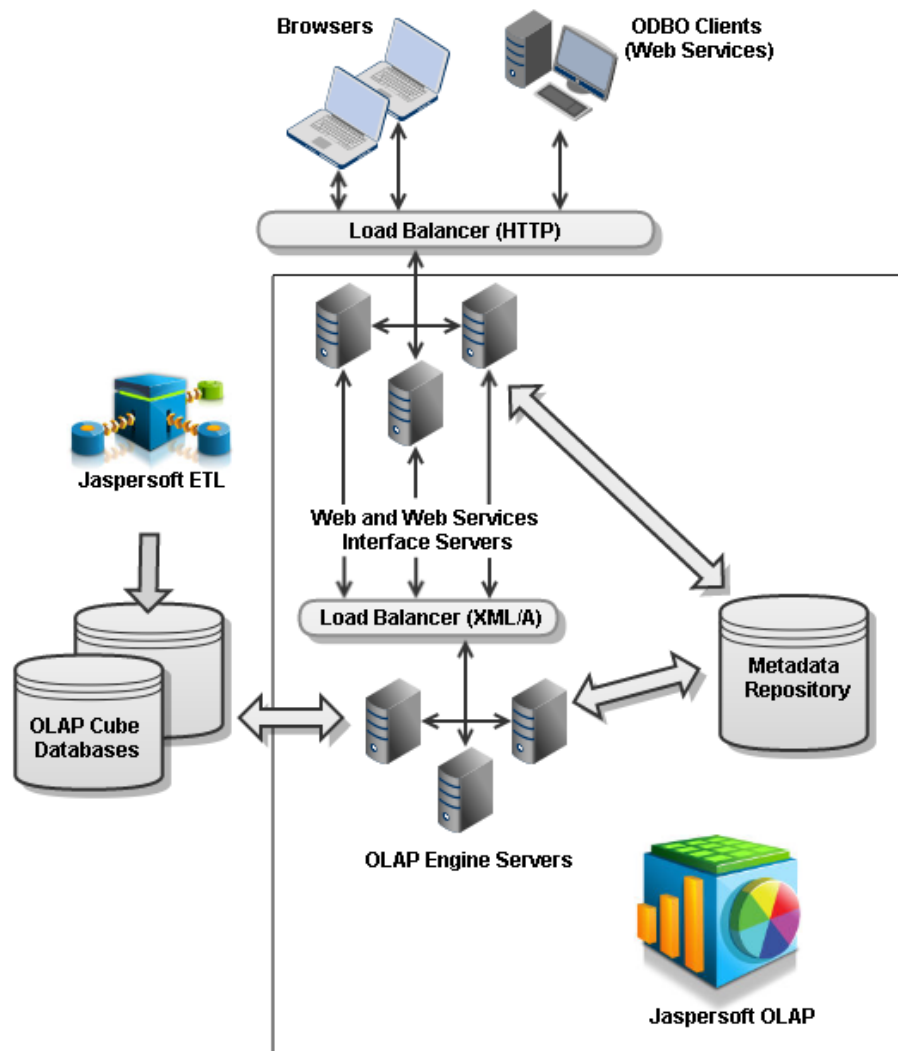
For example, consider the scenario where there are to be two front-ends and two back-ends. If there are sufficient resources, you can put each front-end or engine on a separate computer. If you only have two computers, you can put one front-end on each of the two, and one back-end on each of the two. Users are 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 us 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 the 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 that 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.

[Load-Balanced Jaspersoft OLAP Environment](#) depicts a load-balanced OLAP environment.



*Figure 53: Load-Balanced Jaspersoft OLAP Environment*

## 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 usernames 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 the availability and consistency of the data. Availability, for the purposes of this discussion, has to do with how long it takes to see new data in the reports. Consistency refers to whether 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 us 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 us say it 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 are 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 is consistent with the data. For large amounts of data or numbers of users, this is not recommended, because the performance of viewing uncached reports with every request is 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 [Clearing the Cache](#)) after the new data has been loaded and precache 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 to 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.

## Troubleshooting Information for Technical Support

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

- Product name, edition, and version (for example, JasperReports Server Enterprise 9.0.0). Click About JasperReports Server on any page of the application to view these details. If no edition or license details are shown in the dialog, you are using the Community edition of the server.
- 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, are relevant parts.

- The MDX query that illustrates the error.

Technical Support may request that you use the log collector feature to supply your logs to them so they can understand what may be causing any issues you may encounter. For more information, see the JasperReports Server Administrator Guide.

# Glossary

---

## **Ad Hoc Editor**

The interactive data explorer in JasperReports Server Professional and Enterprise editions. Starting from a predefined collection of fields, the Ad Hoc Editor lets you drag and drop fields, dimensions, and measures to explore data and create tables, charts, and crosstabs. These Ad Hoc views can be saved as reports.

## **Ad Hoc Report**

In previous versions of JasperReports Server, a report was created through the Ad Hoc Editor. Such reports could be added to dashboards and be scheduled, but when edited in Jaspersoft Studio, lost their grouping and sorting. In the current version, the Ad Hoc Editor is used to explore views that in turn can be saved as reports. Such reports can be edited in Jaspersoft Studio without loss, and can be scheduled and added to dashboards.

## **Ad Hoc View**

A view of data that is based on a Domain, Topic, or OLAP client connection. An Ad Hoc view can be a table, chart, or crosstab and is the entry point to analysis operations such as slice and dice, drill down, and drill through. Compare [OLAP View](#) You can save an Ad Hoc view as a report to edit it in the interactive viewer, schedule it, or add it to a dashboard.

## **Aggregate Function**

An aggregate function is one that is computed using a group of values; for example, Sum or Average. Aggregate functions can be used to create calculated fields in Ad Hoc views. Calculated fields containing aggregate functions cannot be used as fields or added to groups in an Ad Hoc view and should not be used as filters. Aggregate functions allow you to set a level, which specifies the scope of the calculation; level values include Current (not available for PercentOf), ColumnGroup, ColumnTotal, RowGroup, RowTotal, Total.

## **Amazon Web Services (AWS)**

Cloud platform, used to provide and host a family of services, such as RDS, S3, and EC2.

## **Analysis View**

See [OLAP View](#).

**Audit Archiving**

To prevent audit logs from growing too large to be easily accessed, the 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 repository 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 an Ad Hoc view or a Domain, a field whose value is calculated from a user-defined formula that may include any number of fields, operators, and constants. For Domains, a calculated field becomes one of the items to which the Domain's security file and locale bundles can apply. There are more functions available for Ad Hoc view calculations than for Domains.

**CloudFormation (CF)**

Amazon Web Services CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning, and updating them in an orderly and predictable fashion.

**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 OLAP 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 OLAP 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 parametrize 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.

**Dashlet**

An element in a dashboard. Dashlets are defined by editable properties that vary depending on the dashlet type. Types of dashlet include reports, text elements, filters, and external web content.

**Data Island**

A single join tree or a table without joins in a Domain. A Domain may contain several data islands, but when creating an Ad Hoc view from a Domain, you can only select one of them to be available in the view.

**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 > Server > Settings Ad Hoc Settings. By default, this setting is only available to the superuser account.

**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 of 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.

**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.

**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 an element of an OLAP 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 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**

In Jaspersoft Studio, a frame is a rectangular element that can contain other elements and optionally draw a border around them. Elements inside a frame are positioned relative to the frame, not to the band, and when you move a frame, all the elements contained in the frame move together. A frame automatically stretches to fit its contents.



## 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 an OLAP cube, a member of a dimension containing a group of members.

## Input Control

A button checkbox, dropdown 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.

## 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.

## JasperReport

A combination of a report template and data that produces a complex document for viewing, printing, or archiving information. In the server, a JasperReport references other resources in the repository:

- The report template (in the form of a JRXML file)
- Information about the data source that supplies data for the report
- Any additional resources, such as images, fonts, and resource bundles referenced by the report template.

The collection of all the resources that are referenced in a JasperReport is sometimes called a report unit. End users usually see and interact with a JasperReport as a single resource in the repository, but report creators must define all of the components in the report unit.

**JasperReports IO**

An HTTP-based reporting service for JasperReports Library that provides a REST API for running, exporting, and interacting with reports and a JavaScript API for embedding reports and their input controls into your web pages and web applications.

**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 Studio**

A commercial open source tool for graphically designing reports that leverage all features of the JasperReports Library. Jaspersoft Studio lets you drag and drop fields, charts, and sub-reports onto a canvas, and also define parameters or expressions for each object to create pixel-perfect reports. You can generate the JRXML of the report directly in Jaspersoft Studio, or upload it to JasperReports Server. Jaspersoft Studio is implemented in Eclipse.

**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.

**Jaspersoft Studio**

An open source tool for graphically designing reports that leverage all features of the JasperReports Library. Jaspersoft Studio lets you drag and drop fields, charts, and sub-

reports onto a canvas, and also define parameters or expressions for each object to create pixel-perfect reports. You can generate the JRXML of the report directly in Jaspersoft Studio, or upload it to JasperReports Server. Jaspersoft Studio is implemented in Eclipse.

**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 a 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 Jaspersoft Studio and JasperReports Server. JRXML is an open format that uses the XML standard to define precisely all the structure and configuration of a report.

**Level**

Specifies the scope of an aggregate function in an Ad Hoc view. Level values include Current (not available for PercentOf), ColumnGroup, ColumnTotal, RowGroup, RowTotal, Total.

**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 OLAP 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 OLAP 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 OLAP client connection that consists of an OLAP schema and a data source. OLAP client connections populate OLAP views.

### **Mondrian Schema Editor**

An open source Eclipse plug-in for creating Mondrian OLAP schemas.

### **Mondrian XML/A Source**

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

## **MySQL**

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

## **Navigation Table**

The main table in an OLAP 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.

**OLAP Client Connection**

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

**OLAP Schema**

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

**OLAP View**

Also called an analysis view. A view of multidimensional data that is based on an OLAP client connection and an MDX query. Unlike Ad Hoc views, you can directly edit an OLAP view's MDX query to change the data and the way they are displayed. An OLAP view is the entry point for advanced analysis users who want to write their own queries. [Compare Ad Hoc View](#).

**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 suborganizations 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 help desk 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.

## Report

In casual usage, a report may refer to:

- A JasperReport. See [JasperReport](#).
- The main JRXML in a JasperReport.
- The file generated when a JasperReport is scheduled. Such files are also called content resources or output files.
- The file generated when a JasperReport is run and then exported.
- In previous JasperReports Server versions, a report is created in the Ad Hoc Editor. See [Ad Hoc Report](#).

## Report Run

An execution of a report, Ad Hoc view, or dashboard, or a view or dashboard designer session, it measures and limits usage of Freemium instances of JasperReports Server. The executions apply to resources no matter how they are run (either in the web interface or through the various APIs, such as REST web services). Users of our Community Project and our full-use commercial licenses are not affected by the limit. For more information, please contact [sales@jaspersoft.com](mailto:sales@jaspersoft.com).

## Repository

Depending on the context:

- In JasperReports Server, the repository is the tree structure of folders that contain all saved reports, dashboards, OLAP views, and resources. Users can access the repository through the JasperReports Server web interface or through Jaspersoft Studio. Applications can access the repository through the web service API. Administrators use the import and export utilities to back up the repository contents.
- In JasperReports IO, the repository is where all the resources needed to create and run reports are stored. The repository can be stored in a directory on the host computer or in an S3 bucket hosted by Amazon Web Services. Users can access the repository through a file browser on the host machine or through the AWS console.

## 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 OLAP 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.

## S3 Bucket

Cloud storage system for Amazon Web Services. JasperReports IO can use an S3 bucket to store files for its repository.

## 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 OLAP 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 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.

**Stack**

A collection of Amazon Web Services resources you create and delete as a single unit.

**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 opens the Ad Hoc Editor.

**Transactional Data**

Data that describes 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.

**View**

Several meanings pertain to JasperReports Server:

- An Ad Hoc view. See [Ad Hoc View](#).
- An OLAP view. See [OLAP View](#).
- A database view. See [http://en.wikipedia.org/wiki/View\\_%28database%29](http://en.wikipedia.org/wiki/View_%28database%29).

**Virtual Data Source**

A virtual data source allows you to combine data residing in multiple JDBC and/or JNDI data sources into a single data source that can query the combined data. Once you have created a virtual data source, you create Domains that join tables across the data sources to define the relationships between the data sources.

**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 the Simple Object Access protocol (SOAP) to access remote data sources. For more information, see <http://www.xmla.org/>.

## **XML/A Connection**

A type of OLAP client connection that consists of Simple Object Access Protocol (SOAP) definitions used to access data on a remote server. OLAP client connections populate OLAP views.

# Jaspersoft Documentation and Support Services

---

For information about this product, you can read the documentation, contact Support, and join Jaspersoft Community.

## How to Access Jaspersoft Documentation

Documentation for Jaspersoft products is available on the [Product Documentation website](#), mainly in HTML and PDF formats.

The [Product Documentation website](#) is updated frequently and is more current than any other documentation included with the product.

## Product-Specific Documentation

The documentation for this product is available on the [JasperReports® Product Documentation](#) page.

## How to Access Related Third-Party Documentation

When working with JasperReports®, you may find it useful to read the documentation of the following third-party products:

## How to Contact Support for Jaspersoft Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our [product Support website](#).
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the [product Support website](#). If you do not have a username, you can request one by clicking **Register** on the website.

## How to Join Jaspersoft Community

Jaspersoft Community is the official channel for Jaspersoft customers, partners, and employee subject matter experts to share and access their collective experience. Jaspersoft Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from Jaspersoft products. In addition, users can submit and vote on feature requests from within the [Jaspersoft Ideas Portal](#). For a free registration, go to [Jaspersoft Community](#).

# Legal and Third-Party Notices

---

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

Jaspersoft, JasperReports, Visualize.js, and TIBCO are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (<https://www.cloud.com/legal>) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at <https://www.tibco.com/patents>.

Copyright © 2005-2024. Cloud Software Group, Inc. All Rights Reserved.