



The Pentaho Analysis Guide



This document is copyright © 2011 Pentaho Corporation. No part may be reprinted without written permission from Pentaho Corporation. All trademarks are the property of their respective owners.

Help and Support Resources

If you have questions that are not covered in this guide, or if you would like to report errors in the documentation, please contact your Pentaho technical support representative.

Support-related questions should be submitted through the Pentaho Customer Support Portal at <http://support.pentaho.com>.

For information about how to purchase support or enable an additional named support contact, please contact your sales representative, or send an email to sales@pentaho.com.

For information about instructor-led training on the topics covered in this guide, visit <http://www.pentaho.com/training>.

Limits of Liability and Disclaimer of Warranty

The author(s) of this document have used their best efforts in preparing the content and the programs contained in it. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, express or implied, with regard to these programs or the documentation contained in this book.

The author(s) and Pentaho shall not be liable in the event of incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of the programs, associated instructions, and/or claims.

Trademarks

Pentaho (TM) and the Pentaho logo are registered trademarks of Pentaho Corporation. All other trademarks are the property of their respective owners. Trademarked names may appear throughout this document. Rather than list the names and entities that own the trademarks or insert a trademark symbol with each mention of the trademarked name, Pentaho states that it is using the names for editorial purposes only and to the benefit of the trademark owner, with no intention of infringing upon that trademark.

Company Information

Pentaho Corporation
Citadel International, Suite 340
5950 Hazeltine National Drive
Orlando, FL 32822
Phone: +1 407 812-OPEN (6736)
Fax: +1 407 517-4575
<http://www.pentaho.com>

E-mail: communityconnection@pentaho.com

Sales Inquiries: sales@pentaho.com

Documentation Suggestions: documentation@pentaho.com

Sign-up for our newsletter: <http://community.pentaho.com/newsletter/>

Contents

Introduction.....	6
ROLAP Defined.....	6
Workflow Overview.....	6
Pentaho Analysis Enterprise Edition Features.....	7
Preparing Your Data.....	8
Dimensional Modeling.....	9
Virtual ROLAP Cubes.....	9
Prototyping With Pentaho Data Integration.....	11
Creating a Prototype Schema With a Non-PDI Data Source.....	11
Creating a Prototype Schema With a PDI Data Source.....	11
Testing With Pentaho Analyzer and Report Wizard.....	12
Prototypes in Production.....	12
Creating and Modifying Mondrian Schemas.....	13
Schema Workbench Notes.....	13
Adding a Data Source in Schema Workbench.....	13
Adding a Simple JNDI Data Source For Design Tools.....	14
Adding a JDBC Driver.....	14
Removing Mondrian Data Sources.....	15
Restricting Schema Access in the Pentaho User Console.....	16
Creating a Mondrian Schema.....	16
Using MDX Mode and Hand-Editing a Schema.....	17
Adding Business Groups.....	18
Adding Field Descriptions.....	18
Building a Schema and Detecting Errors.....	18
Restricting Access to Specific Members.....	18
Adapting Mondrian Schemas to Work with Pentaho Analyzer.....	19
Applying Relative Date Filters.....	19
Adjusting for Calculated Members and Named Sets.....	20
Localization/Internationalization of Analysis Schemas.....	20
Replacing Or Translating BI Server Message Bundles.....	21
Pentaho Analyzer Localization.....	22
Setting a Default Font for PDF Exports.....	22
Configuring Analysis Options.....	23
Managing Analysis Data Sources.....	23
Configuring the Mondrian Engine (mondrian.properties).....	23
Configuring Pentaho Analyzer (analyzer.properties).....	24
Adjusting Drill Link Limitations.....	24
PDF Logo and Cover Page Customization.....	24
Setting Query Limits.....	25
Divide by Zero Display.....	25
Defining Global Default Chart Options in Analyzer.....	25
Mondrian Cache Control.....	27
Segment Cache Architecture.....	27
Cache Configuration Files.....	29
Modifying the JGroups Configuration.....	29
Switching to Another Cache Framework.....	29
Switching to Memcached.....	30
Switching to Pentaho Platform Delegating Cache.....	31
Using a Custom SegmentCache SPI.....	31
Clearing the Mondrian Cache.....	31
Priming the Cache.....	31
Mondrian Role Mapping in the BI Server.....	32
The Mondrian-One-To-One-UserRoleMapper.....	32
The Mondrian-SampleLookupMap-UserRoleMapper.....	32

The Mondrian-SampleUserSession-UserRoleMapper.....	32
Visualizing Your Data.....	33
Introduction to the Multidimensional Expression Language (MDX).....	33
Further Reading.....	33
Creating Analyzer Reports and Analysis Views.....	33
Creating a New Analyzer Report.....	34
Using Analyzer on an Apple iPad.....	47
Creating a New Analysis View.....	48
Using Analysis Cubes in Report Designer.....	48
Adding an OLAP Data Source.....	48
Creating an MDX Query.....	49
Logging.....	50
Analysis SQL Output Logging.....	50
Enabling Segment Cache Logging.....	50
Viewing Log Output in Analyzer.....	50
Troubleshooting.....	52
Old Analysis Schemas Still Show Up in Pentaho User Console.....	52
Multi-Byte Characters Don't Appear In PDFs Exported From Analyzer.....	52
Mondrian Schema Element Quick Reference.....	53
AggExclude.....	54
AggFactCount.....	55
AggForeignKey.....	55
AggIgnoreColumn.....	55
AggLevel.....	55
AggMeasure.....	55
AggName.....	56
AggPattern.....	56
AggTable.....	56
CalculatedMember.....	56
CalculatedMemberProperty.....	57
CaptionExpression.....	57
Closure.....	58
ColumnDef.....	58
ColumnDefs.....	58
Cube.....	58
CubeGrant.....	59
CubeUsage.....	59
CubeUsages.....	59
Dimension.....	60
DimensionGrant.....	60
DimensionUsage.....	60
Formula.....	61
Hierarchy.....	61
HierarchyGrant.....	62
InlineTable.....	62
Join.....	63
KeyExpression.....	63
Level.....	63
Measure.....	65
MeasureExpression.....	66
MemberGrant.....	66
NamedSet.....	66
NameExpression.....	67
OrdinalExpression.....	67
Parameter.....	67
ParentExpression.....	67
Property.....	68
PropertyExpression.....	68
Role.....	68
RoleUsage.....	68

Row.....	69
Rows.....	69
Schema.....	69
SchemaGrant.....	69
SQL.....	70
Table.....	70
Union.....	71
UserDefinedFunction.....	71
Value.....	71
View.....	71
VirtualCube.....	71
VirtualCubeDimension.....	72
VirtualCubeMeasure.....	72

Introduction

This guide helps BI administrators prepare their data for use with the Pentaho Analyzer, Pentaho Report Designer, and JPivot client tools. The collection of analysis components in the Pentaho BI Suite (referred to generally as Pentaho Analysis) enables you to visualize data trends and reveal useful information about your business. You can do this by creating static reports from an analysis data source, traversing an analysis cube through an Analyzer report, showing how data points compare by using charts, and monitor the status of certain trends and thresholds with dashboards.

Before you can begin using any client tools, you must consolidate data from disparate sources into one canonical source and optimize it for the metrics you want to analyze; create an analysis schema to describe the data; iteratively improve that schema so that it meets your users' needs; and create aggregation tables for frequently computed views.

ROLAP Defined

Pentaho Analysis is built on the Mondrian relational online analytical processing (**ROLAP**) engine. ROLAP relies on a multidimensional data model that, when queried, returns a dataset that resembles a grid. The rows and columns that describe and bring meaning to the data in that grid are **dimensions**, and the hard numerical values in each cell are the **measures** or **facts**. In Pentaho Analyzer, dimensions are shown in yellow and measures are in blue.

ROLAP requires a properly prepared data source in the form of a star or snowflake schema that defines a logical multi-dimensional database and maps it to a physical database model. Once you have your initial data structure in place, you must design a descriptive layer for it in the form of a Mondrian schema, which consists of one or more cubes, hierarchies, and members. Only when you have a tested and optimized Mondrian schema is your data prepared on a basic level for end-user tools like Pentaho Analyzer and JPivot. See [Workflow Overview](#) on page 6 for a more comprehensive overview of the Pentaho Analysis data preparation workflow, including which Pentaho tools you will need to execute this process.

For concise definitions of ROLAP terms, refer to [Mondrian Schema Element Quick Reference](#) on page 53 and the individual element pages it references.

Workflow Overview

To prepare data for use with the Pentaho Analysis (and Reporting, to a certain extent) client tools, you should follow this basic workflow:

Design a Star or Snowflake Schema

The entire process starts with a data warehouse. This guide will not attempt to explain how to build this structure -- there are entire books on the subject, and an entire consulting industry dedicated to it already. The end result should be data model in the star or snowflake schema pattern. You don't have to worry too much about getting the model exactly right on your first try. Just cover all of your anticipated business needs; part of the process is coming back to the data warehouse design step and making changes to your initial data model after you've discovered what your operational needs are.

Populate the Star/Snowflake Schema

Once your data model is designed, the next step is to populate it with actual data, thereby creating your data warehouse. The best tool for this job is Pentaho Data Integration, an enterprise-grade extract, transform, and load (ETL) application.

Build a Mondrian Schema

Now that your initial data warehouse project is complete, you must build a Mondrian schema to organize and describe it in terms that Pentaho Analysis can understand. This is also accomplished through Pentaho Data Integration by using the Agile BI plugin. Just connect to your data warehouse and auto-populate your schema with the Modeler graphical interface. Alternatively, you can use Pentaho Schema Workbench to create an analysis schema through a manual process.

Initial Testing

At this point you should have a multi-dimensional data structure with an appropriate metadata layer. You can now start using Pentaho Analyzer and JPivot to drill down into your data and see if your first attempt at data modelling was successful. In all likelihood, it will need some adjustment, so take note of all of the schema limitations that you're unhappy with during this initial testing phase.

Do not be concerned with performance issues at this time -- just concentrate on the completeness and comprehensiveness of the data model.

Adjust and Repeat Until Satisfied

Use the notes you took during the testing phase to redesign your data warehouse and Mondrian schema appropriately. Adjust hierarchies and relational measure aggregation methods. Create virtual cubes for analyzing multiple fact tables by conforming dimensions. Re-test the new implementation and continue to refine the data model until it matches your business needs perfectly.

Test for Performance

Once you're satisfied with the design and implementation of your data model, you should try to find performance problems and address them by tuning your data warehouse database, and by creating aggregation tables. The testing can only be reasonably done by hand, using Pentaho Analyzer and/or JPivot. Take note of all of the measures that take an unreasonably long time to calculate. Also, enable SQL logging and locate slow-performing queries, and build indexes for optimizing query performance.

Create Aggregation Tables

Using your notes as a guide, create aggregation tables in Pentaho Aggregation Designer to store frequently computed analysis views. Re-test and create new aggregation tables as necessary.

If you are working with a relatively small data warehouse or a limited number of dimensions, you may not have a real need for aggregation tables. However, be aware of the possibility that performance issues may come up in the future. Check in with your users occasionally to see if they have any particular concerns about the speed of their BI content.

Deploy to Production

Your data warehouse and Mondrian schema have been created, tested, and refined. You're now ready to put it all into production. You may need to personally train or purchase Pentaho training for anyone in your organization who needs to create traditional reports, dashboards, or Analyzer reports with Pentaho's client tools.

Pentaho Analysis Enterprise Edition Features

Pentaho offers expanded functionality for Pentaho Analysis Enterprise Edition customers:

- The Pentaho Analyzer visualization tool
- A pluggable Enterprise Cache with support for highly scalable, distributable cache implementations including Infinispan, Memcached, and Terracotta BigMemory
- Greenplum dialect
- Hive dialect

Use of these features requires a Pentaho Analysis Enterprise Edition license installed on the BI Server and workstations that have Schema Workbench and Metadata Editor. A special BI Server package must also be installed; this process is covered in the Pentaho BI Suite installation documentation.

All relevant configuration options for these features are covered in this guide.

Preparing Your Data

This section deals primarily with Pentaho Data Integration (the Kettle project), a graphical tool that helps you consolidate many data sources into one. Because data warehouse and ROLAP schema design can vary wildly among implementations, there is no specific advice on them here.

Consult the *Pentaho Data Integration User Guide* and *Pentaho Data Integration Administrator's Guide* for more information about extract, transform, and load (ETL) processes for data warehouse creation.

Dimensional Modeling

Dimensional modeling is the process of transforming data from multiple sources in non-human-friendly formats into a single data source that is organized to support business analytics. The workflow for developing a dimensional model is along these lines:

1. Collect user requirements for business logic and processes
2. Considering the entirety of your data, break it down into subjects
3. Isolate groups of facts into one or more fact tables
4. Design dimensional tables that draw relationships between levels (fact groups)
5. Determine which members of each level are useful for each dimensional table
6. Build and publish a Mondrian (Pentaho Analysis) schema and collect feedback from users
7. Refine your model based on user feedback, continue iterating through this list until users are productive


Or, expressed as a series of questions:

1. What topics or subjects are important to the users who are analyzing the data? What do your users need to learn from the data?
2. What are the important details your users will need to examine in the data?
3. How should each data column relate to other data columns?
4. How should datasets be grouped and organized?
5. What are some useful short descriptions for each dimensional level in a hierarchy (for each element, decide what is useful within that element; for instance, in a dimensional table representing time, your levels might be year, month, and day, and your members for the year level might be 2003, 2004, 2005).
6. How effective is this dimensional model for the intended userbase? How can it improve?

The Agile BI tools in Pentaho Data Integration make dimensional modeling much easier than the traditional methods. Through PDI, you can quickly adjust your business logic, the granularity of your fact tables, and the attributes of your dimension tables, then generate a new model and push it out to a test environment for evaluation.

Virtual ROLAP Cubes


Another name for a dimensional model is a **cube**. Each cube represents one fact table and several dimensional tables. This model should be useful for reporting and analysis on the subject of the data in the fact table. However, if you want to cross-reference this data with another cube -- if you need to analyze data across two or more cubes, or need to combine information from two fact tables on the same subject but with different granularity -- then you must create a **virtual cube**. The XML elements that compose a virtual cube are explained in detail below.

 **Note:** Virtual cubes cannot presently be created through Pentaho Data Integration's model perspective; you must use Schema Workbench instead.

The **<CubeUsages>** element specifies the cubes that are imported into the virtual cube. It holds **<CubeUsage>** elements.

The **<CubeUsage>** element specifies the base cube that is imported into the virtual cube. Alternatively you can define a **<VirtualCubeMeasure>** and use similar imports from the base cube without defining a **<CubeUsage>**. The **cubeName** attribute specifies the name of the base cube. The **ignoreUnrelatedDimensions** attribute determines whether or not the measures from this base cube will have non-joining dimension members pushed to the top level member. This attribute is false by default because it is still experimental.

The **<VirtualCubeDimension>** element imports a dimension from one of the constituent cubes. If you do not specify the **cubeName** attribute, this means you are importing a shared dimension.

 **Note:** If a shared dimension is used more than once in a cube, there is no way to determine which usage of the shared dimension you intend to import.

The **<VirtualCubeMeasure>** element imports a measure from one of the constituent cubes. It is imported with the same name. If you want to create a formula or rename a measure as you import it, use the **<CalculatedMember>** element instead.

Virtual cubes are useful for situations where there are fact tables of different granularities (for instance, one Time fact table might be configured on a Day level, another at the Month level), or fact tables of different dimensionalities (for

instance one on Products, Time and Customer, another on Products, Time and Warehouse), and you need to present the results to users who don't know how the data is structured.

Any common dimensions -- shared dimensions which are used by both constituent cubes -- are automatically synchronized. In this example, [Time] and [Products] are common dimensions. So if the context is ([Time].[2005].[Q2], [Products].[Productname].[P-51-D Mustang]), measures from either cube will relate to this context.

Dimensions which only belong to one cube are called **non-conforming dimensions**. The [Gender] dimension is an example of this; it exists in the Sales cube, but not Warehouse. If the context is ([Gender].[F], [Time].[2005].[Q1]), it makes sense to ask the value of the [Unit Sales] measure (which comes from the [Sales] cube) but not the [Units Ordered] measure (from [Warehouse]). In the context of [Gender].[F], [Units Ordered] has value NULL.

```
<VirtualCube name="Warehouse and Sales">
  <CubeUsages>
    <CubeUsage cubeName="Sales" ignoreUnrelatedDimensions="true"/>
    <CubeUsage cubeName="Warehouse"/>
  </CubeUsages>
  <VirtualCubeDimension cubeName="Sales" name="Customers"/>
  <VirtualCubeDimension cubeName="Sales" name="Education Level"/>
  <VirtualCubeDimension cubeName="Sales" name="Gender"/>
  <VirtualCubeDimension cubeName="Sales" name="Marital Status"/>
  <VirtualCubeDimension name="Product"/>
  <VirtualCubeDimension cubeName="Sales" name="Promotion Media"/>
  <VirtualCubeDimension cubeName="Sales" name="Promotions"/>
  <VirtualCubeDimension name="Store"/>
  <VirtualCubeDimension name="Time"/>
  <VirtualCubeDimension cubeName="Sales" name="Yearly Income"/>
  <VirtualCubeDimension cubeName="Warehouse" name="Warehouse"/>
  <VirtualCubeMeasure cubeName="Sales" name="[Measures].[Sales Count]"/>
  <VirtualCubeMeasure cubeName="Sales" name="[Measures].[Store Cost]"/>
  <VirtualCubeMeasure cubeName="Sales" name="[Measures].[Store Sales]"/>
  <VirtualCubeMeasure cubeName="Sales" name="[Measures].[Unit Sales]"/>
  <VirtualCubeMeasure cubeName="Sales" name="[Measures].[Profit
Growth]"/>
  <VirtualCubeMeasure cubeName="Warehouse" name="[Measures].[Store
Invoice]"/>
  <VirtualCubeMeasure cubeName="Warehouse" name="[Measures].[Supply
Time]"/>
  <VirtualCubeMeasure cubeName="Warehouse" name="[Measures].[Units
Ordered]"/>
  <VirtualCubeMeasure cubeName="Warehouse" name="[Measures].[Units
Shipped]"/>
  <VirtualCubeMeasure cubeName="Warehouse" name="[Measures].[Warehouse
Cost]"/>
  <VirtualCubeMeasure cubeName="Warehouse" name="[Measures].[Warehouse
Profit]"/>
  <VirtualCubeMeasure cubeName="Warehouse" name="[Measures].[Warehouse
Sales]"/>
  <VirtualCubeMeasure cubeName="Warehouse" name="[Measures].[Average
Warehouse Sale]"/>
  <CalculatedMember name="Profit Per Unit Shipped" dimension="Measures">
    <Formula>[Measures].[Profit] / [Measures].[Units Shipped]</
Formula>
  </CalculatedMember>
</VirtualCube>
```


Prototyping With Pentaho Data Integration

Pentaho Data Integration offers rapid prototyping of analysis schemas through a mix of processes and tools known as **Agile BI**. The Agile BI functions of Pentaho Data Integration are explained in this section, but there is no further instruction here regarding PDI installation, configuration, or use beyond ROLAP schema creation. If you need information related to PDI in general, consult the *Pentaho Data Integration Installation Guide* and/or the *Pentaho Data Integration User Guide* in the Pentaho Knowledge Base.

Creating a Prototype Schema With a Non-PDI Data Source

Your data sources must be configured, running, and available before you can proceed with this step.

Follow the below procedure to create a ROLAP schema prototype from an existing database, file, or data warehouse.

 **Note:** If you are already using PDI to create your data source, skip these instructions and refer to [Creating a Prototype Schema With a PDI Data Source](#) on page 11 instead.

1. Start Spoon and connect to your repository, if you are using one.

```
cd ~/pentaho/design-tools/data-integration/ && ./spoon.sh
```

2. Go to the **File** menu, then select the **New** sub-menu, then click on **Model**.

The interface will switch over to the **Model** perspective.

3. In the **Properties** pane on the right, click **Select**.

A data source selection window will appear.

4. Click the round green **+** icon in the upper right corner of the window.

The **Database Connection** dialogue will appear.

5. Enter in and select the connection details for your data source, then click **Test** to ensure that everything is correct. Click **OK** when you're done.

6. Select your newly-added data source, then click **OK**.

The **Database Explorer** will appear.

7. Traverse the database hierarchy until you get to the table you want to create a model for. Right-click the table, then select **Model** from the context menu.

The Database Explorer will close and bring you back to the Model perspective.

8. Drag items from the **Data** pane on the left and drop them into either the **Measures** or **Dimensions** groups in the **Model** pane in the center.

The Measures and Dimensions groups will expand to include the items you drag into them.

9. Select each new measure and dimension item, and modify its details accordingly in the **Properties** pane on the right.

10. Save your model through the **File** menu, or publish it to the BI Server using the **Publish** icon above the Model pane.

You now have a basic ROLAP schema. You should test it yourself before putting it into production. To do this, continue on to [Testing With Pentaho Analyzer and Report Wizard](#) on page 12.

Creating a Prototype Schema With a PDI Data Source

Context for the current task

1. Start Spoon and connect to your repository, if you are using one.

```
cd ~/pentaho/design-tools/data-integration/ && ./spoon.sh
```

2. Open the transformation that produces the data source you want to create a ROLAP schema for.

3. Right-click your output step, then select **Model** from the context menu.

4. Drag items from the **Data** pane on the left and drop them into either the **Measures** or **Dimensions** groups in the **Model** pane in the center.

The Measures and Dimensions groups will expand to include the items you drag into them.

5. Select each new measure and dimension item, and modify its details accordingly in the **Properties** pane on the right.
6. Save your model through the **File** menu, or publish it to the BI Server using the **Publish** icon above the Model pane.

You now have a basic ROLAP schema. You should test it yourself before putting it into production. To do this, continue on to [Testing With Pentaho Analyzer and Report Wizard](#) on page 12.

Testing With Pentaho Analyzer and Report Wizard

You must have an analysis schema with at least one measure and one dimension, and it must be currently open and focused on the Model perspective in Spoon.

This section explains how to use the embedded Analyzer and Report Design Wizard to test a prototype analysis schema.

1. While in the Model perspective, select your visualization method from the drop-down box above the Data pane (it has a **New:** to its left), then click **Go**.

The two possible choices are: **Pentaho Analyzer** and **Report Wizard**. You do not need to have license keys for Pentaho Analysis or Pentaho Reporting in order to use these preview tools.

2. Either the Report Design Wizard will launch in a new sub-window, or Pentaho Analyzer will launch in a new tab. Use it as you would in Report Designer or the Pentaho User Console.

3. When you have explored your new schema, return to the Model perspective by clicking **Model** in the upper right corner of the Spoon toolbar, where all of the perspective buttons are.

Do not close the tab; this will close the file, and you will have to reopen it in order to adjust your schema.

4. If you continue to refine your schema in the Model perspective, you must click the **Go** button again each time you want to view it in Analyzer or Report Wizard; the Visualize perspective does not automatically update according to the changes you make in the Modeler.

You now have a preview of what your model will look like in production. Continue to refine it through the Model perspective, and test it through the Visualize perspective, until you meet your initial requirements.

Prototypes in Production

Once you're ready to move your analysis schema to production, use the **Publish** button above the Model pane in the Model perspective, and use it to connect to your production BI Server.

You can continue to refine your schema if you like, but it must be republished each time you want to redeploy it.

Creating and Modifying Mondrian Schemas

Now that you have a physical data model in place, you must create a logical model that maps to it. A Mondrian schema is essentially an XML file that performs this mapping, thereby defining a multidimensional database structure.

In a very basic scenario, you will create a Mondrian schema with one cube that consists of a single fact table and a few dimensions, each with a single hierarchy consisting of a handful of levels. More complex schemas may involve multiple virtual cubes, and instead of mapping directly to the single fact table at the center of a star schema, they might map to views or inline tables instead.

All of the Mondrian XML elements are documented in this section in both a single quick reference list and a full individual reference piece for each element. Primarily you will be using Pentaho Schema Workbench to create Mondrian schemas graphically, though advanced schema design may require hand-editing of the bare XML later on; this is also possible within the Schema Workbench interface.

Schema Workbench Notes

Before you start using Schema Workbench, you should be aware of the following points:

- You start Schema Workbench by executing the `/pentaho/design-tools/schema-workbench/workbench` script. On Linux and OS X, this is a `.sh` file; on Windows it's `.bat`.
- You must be familiar with your physical data model before you use Schema Workbench. If you don't know which are your fact tables and how your dimensions relate to them, you will not be able to make significant progress in developing a Mondrian schema.
- When you make a change to any field in Schema Workbench, the change will not be applied until you click out of that field such that it loses the cursor focus.
- Schema Workbench is designed to accommodate multiple sub-windows. By default they are arranged in a cascading fashion. However, you may find more value in a tiled format, especially if you put the JDBC Explorer window next to your Schema window so that you can see the database structure at a glance. Simply resize and move the sub-windows until they are in agreeable positions.

Adding a Data Source in Schema Workbench

Your data source must be available, its database driver JAR must be present in the `/pentaho/design-tools/schema-workbench/drivers/` directory, and you should know or be able to obtain the database connection information and user account credentials for it.


Follow the below process to connect to a data source in Schema Workbench.


1. Establish a connection to your data source by going to the **Options** menu and selecting **Connection**.

The **Database Connection** dialogue will appear.

2. Select your database type, then enter in the necessary database connection information, then click **Test**. When you've verified that the connection settings work, click **OK**.

The database connection information includes the database name, port number, and user credentials. If you don't know what to type into any of these fields, consult your database administrator or database vendor's documentation.

 **Note:** The **Require Schema** checkbox, when selected in the Options menu, puts Schema Workbench into a mode where unpopulated elements appear in the schema.

 **Note:** If you are using an Oracle data source, selecting Require Schema will dramatically improve your Analysis schema load time.

3. If you required a database schema in the previous step, you must now define it by going to the **Options** section of the database dialogue, and creating a parameter called **FILTER_SCHEMA_LIST** with a value of the schema name you want to use.

Your data is now available to Schema Workbench, and you can proceed with creating a Mondrian schema.

Adding a Simple JNDI Data Source For Design Tools

Pentaho provides a method for defining a JNDI connection that exists only for locally-installed client tools. This is useful in scenarios where you will be publishing to a BI Server that has a JNDI data source; if you establish the same JNDI connection on your client tool workstation, you will not need to change any data source details after publishing to the BI Server. Follow the directions below to establish a simple JNDI connection for Pentaho client tools.

1. Navigate to the **.pentaho** directory in your home or user directory.

For a user name of **fbeuller**, typically in Linux and Solaris this would be `/home/fbeuller/.pentaho/`, and in Windows it would be `C:\Users\fbeuller\.pentaho\`

2. Switch to the `~/ .pentaho/simple-jndi/` subdirectory. If it does not exist, create it.
3. Edit the **default.properties** file found there. If it does not exist, create it now.

4. Add a data source by declaring a JNDI name followed by a forward slash, then a JNDI parameter and its proper value.

Refer to [Simple JNDI Options](#) on page 14 for more information on parameter options.

```
SampleData/type=javax.sql.DataSource
SampleData/driver=org.hsqldb.jdbcDriver
SampleData/user=pentaho_user
SampleData/password=password
SampleData/url=jdbc:hsqldb:mem:SampleData
```

5. Save and close the file.

You now have a global Pentaho data source that can be used across all of the client tools installed on this machine. You must restart any running Pentaho program in order for this change to take effect.

Simple JNDI Options


Each line in the data source definition must begin with the JNDI name and a forward slash (/), followed by the required parameters listed below.

Parameter	Values
type	javax.sql.DataSource defines a JNDI data source type.
driver	This is the driver class name provided by your database vendor.
user	A user account that can connect to this database.
password	The password for the previously declared user.
url	The database connection string provided by your database vendor.

```
SampleData/type=javax.sql.DataSource
SampleData/driver=org.hsqldb.jdbcDriver
SampleData/user=pentaho_user
SampleData/password=password
SampleData/url=jdbc:hsqldb:mem:SampleData
```

Adding a JDBC Driver

Before you can connect to a data source in any Pentaho server or client tool, you must first install the appropriate database driver. Your database administrator, CIO, or IT manager should be able to provide you with the proper driver JAR. If not, you can download a JDBC driver JAR file from your database vendor or driver developer's Web site. Once you have the JAR, follow the instructions below to copy it to the driver directories for all of the BI Suite components that need to connect to this data source.


 **Note:** Microsoft SQL Server users frequently use an alternative, non-vendor-supported driver called JTDS. If you are adding an MSSQL data source, ensure that you are installing the correct driver.

Backing up old drivers


You must also ensure that there are no other versions of the same vendor's JDBC driver installed in these directories. If there are, you may have to back them up and remove them to avoid confusion and potential class loading problems. This is of particular concern when you are installing a driver JAR for a data source that is the same database type as your Pentaho solution repository. If you have any doubts as to how to proceed, contact your Pentaho support representative for guidance.

Installing JDBC drivers

Copy the driver JAR file to the following directories, depending on which servers and client tools you are using (Dashboard Designer, ad hoc reporting, and Analyzer are all part of the BI Server):

 **Note: For the BI Server:** before copying a new JDBC driver, ensure that there is not a different version of the same JAR in the destination directory. If there is, you must remove the old JAR to avoid version conflicts.

- **BI Server:** `/pentaho/server/biserver-ee/tomcat/lib/`
- **Enterprise Console:** `/pentaho/server/enterprise-console/jdbc/`
- **Data Integration Server:** `/pentaho/server/data-integration-server/tomcat/webapps/pentaho-di/WEB-INF/lib/`
- **Data Integration client:** `/pentaho/design-tools/data-integration/libext/JDBC/`
- **Report Designer:** `/pentaho/design-tools/report-designer/lib/jdbc/`
- **Schema Workbench:** `/pentaho/design-tools/schema-workbench/drivers/`
- **Aggregation Designer:** `/pentaho/design-tools/agg-designer/drivers/`
- **Metadata Editor:** `/pentaho/design-tools/metadata-editor/libext/JDBC/`

 **Note:** To establish a data source in the Pentaho Enterprise Console, you must install the driver in both the Enterprise Console and the BI Server or Data Integration Server. If you are just adding a data source through the Pentaho User Console, you do not need to install the driver to Enterprise Console.

Restarting


Once the driver JAR is in place, you must restart the server or client tool that you added it to.

Connecting to a Microsoft SQL Server using Integrated or Windows Authentication

The JDBC driver supports Type 2 integrated authentication on Windows operating systems through the **integratedSecurity** connection string property. To use integrated authentication, copy the **sqljdbc_auth.dll** file to all the directories to which you copied the JDBC files.

The **sqljdbc_auth.dll** files are installed in the following location:

```
<installation directory>\sqljdbc_<version>\<language>\auth\
```

 **Note:** Use the **sqljdbc_auth.dll** file, in the x86 folder, if you are running a 32-bit Java Virtual Machine (JVM) even if the operating system is version x64. Use the **sqljdbc_auth.dll** file in the x64 folder, if you are running a 64-bit JVM on a x64 processor. Use the **sqljdbc_auth.dll** file in the IA64 folder, you are running a 64-bit JVM on an Itanium processor.

Removing Mondrian Data Sources

Every time you publish a schema from Schema Workbench or Pentaho Data Integration, a new XMLA data source entry is created in `/pentaho/server/biserver-ee/pentaho-solutions/system/olap/datasources.xml`. If you modify a schema and republish it, the data source entry will be updated accordingly. You do not have to manually add anything to this file, and under most circumstances you won't have to modify or remove anything from it, either. However, there is no automatic method to expire a registered data source, so each time you publish a new schema, a new entry is permanently added to `datasources.xml`.

The Pentaho User Console uses `datasources.xml` to populate a schema drop-down selection list that appears when users create new analysis views and Analyzer reports. As you phase out old analysis schemas, you will have to manually remove their `<DataSource>` entries in `datasources.xml`.

Below is the example `datasources.xml` that ships with a default Pentaho configuration:

datasources.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<DataSources>
  <DataSource>
    <DataSourceName>Provider=Mondrian;DataSource=Pentaho</DataSourceName>
    <DataSourceDescription>Pentaho BI Platform Datasources</DataSourceDescription>
    <URL>http://localhost:8080/pentaho/Xmla?userid=joe&password=password</URL>
    <DataSourceInfo>Provider=mondrian</DataSourceInfo>
    <ProviderName>PentahoXMLA</ProviderName>
    <ProviderType>MDP</ProviderType>
    <AuthenticationMode>Unauthenticated</AuthenticationMode>
    <Catalogs>
      <Catalog name="SteelWheels">
        <DataSourceInfo>Provider=mondrian;DataSource=SampleData</DataSourceInfo>
        <Definition>solution:steel-wheels/analysis/
steelwheels.mondrian.xml</Definition>
      </Catalog>
      <Catalog name="SampleData">
        <DataSourceInfo>Provider=mondrian;DataSource=SampleData</DataSourceInfo>
        <Definition>solution:steel-wheels/analysis/SampleData.mondrian.xml</
Definition>
      </Catalog>
    </Catalogs>
  </DataSource>
</DataSources>
```

Restricting Schema Access in the Pentaho User Console

To enable user-level access restrictions for each published analysis schema in the Pentaho User Console, follow the instructions below.

1. Log into the Pentaho User Console as the administrator user.
2. Create new a solution directory or the user you want to provide a private analysis schema for.
3. Change the access permissions on the new user directory so that only the specified user has access to it.
4. Copy the target schema to the private directory that you just created.
5. Rename the copied schema such that it reflects the user account that now owns it.
6. Refresh the solution repository.
7. Edit the `/pentaho/server/biserver-ee/pentaho-solutions/system/olap/datasources.xml` file and add data source entries for each personalized schema copy you created.

```
<Catalog name="Suzy">
  <DataSourceInfo>Provider=mondrian;DataSource=SampleData</DataSourceInfo>
  <Definition>solution:suzy/suzy.mondrian.xml</Definition>
</Catalog>
<Catalog name="Tiffany">
  <DataSourceInfo>Provider=mondrian;DataSource=SampleData</DataSourceInfo>
  <Definition>solution:tiffany/tiffany.mondrian.xml</Definition>
</Catalog>
```

8. Edit the `/pentaho/server/biserver-ee/pentaho-solutions/system/pentaho.xml` file and add a **.xml** to the end of the **acl-files** list.

```
<acl-files>xaction,url,prpt,xdash,xcdf,xanalyzer,xanalyzer.xml</acl-files>
```

9. Restart the BI Server.

The schema copies you created are now only available to the users you specified.

Creating a Mondrian Schema

In order to complete this process, you should have already connected to your data source in Schema Workbench.

This section explains the basic procedure for creating a barebones Mondrian schema using Schema Workbench. If you are confused about the definition or application of any of the elements discussed below, refer to the [Mondrian Schema Element Quick Reference](#) on page 53 and the individual reference pieces it links to.

1. To create a new Mondrian schema, click the New button, or go to the **File** menu, then select **New**, then **Schema**. A new schema sub-window will appear. Resize it to fit your preference.
2. It's easier to visualize your physical data model if you have it in front of you. Turn on the JDBC Explorer from the **New** section of the **File** menu and position it according to your preference. If you have a third-party database visualization tool that you are more familiar with, use that instead.
The JDBC Explorer is not interactive; it only shows the table structure of your data source so that you can see at a glance what the names of the columns and rows in it.
3. Typically your first action when creating a schema is to add a cube. Right-click the Schema icon in the schema window, then select **Add cube** from the context menu. Alternatively you can click the New Cube button in the toolbar.
A new default cube will show up in your schema.
4. Give your cube a name.
5. Add a table by clicking the New Table button, or by right-clicking your cube, then selecting **Add Table**.
This will be your fact table. Alternatively you can select **View** or **Inline Table** if these are the data types you need for your fact table.
6. Click the **Table** entry in the **name** field of your new table, and select or type in the name of the table in your physical model that you want to use for this cube's fact table.
7. Add a dimension by right-clicking the cube, then selecting **Add Dimension**, or by clicking the New Dimension button.
8. Type in a friendly name for this dimension in the **name** field.
9. Select a foreign key for this dimension from the **foreignKey** drop-down box, or just type it into the field.
10. When you add a dimension, a new hierarchy is automatically created for it. To configure the hierarchy, expand the dimension by clicking the lever icon on the left side of the dimension's tree entry, then click on **New Hierarchy 0**. Choose a **primaryKey** or **primaryKey Table**.
11. Add a table to the hierarchy by right-clicking the hierarchy, then selecting **Add Table** from the context menu.
12. Choose a column for the **name** attribute.
13. Add a level to the hierarchy by right-clicking the hierarchy, then selecting **Add Level** from the context menu.
14. Give the level a **name** and choose a **column** for it.
15. Add a *member property* to the level by right-clicking the level, then selecting **Add Property** from the context menu.
16. Give the property a **name** and choose a **column** for it.
17. Add a measure to the cube by right-clicking the cube and selecting **Add Measure** from the context menu.
18. Choose a **column** that you want to provide values for, then select an **aggregator** to determine how the values should be calculated.

These instructions have shown you how to use Schema Workbench's interface to add and configure basic Mondrian schema elements.

When your schema is finished, you should test it with a basic MDX query such as:

```
select {[Dim1].[All Dim1s]} on rows, {[Measures].[Meas1]} on columns from [CubeName]
```

In order to use your schema as a data source in any Pentaho BI Suite client tools, you must publish it to the BI Server. To do this, select **Publish** from the **File** menu, then enter in your BI Server connection information and credentials when requested.

Using MDX Mode and Hand-Editing a Schema

There are two advanced tools in Schema Workbench that enable you to work with raw MDX and XML. The first is the MDX query editor, which can query your logical data model in real time. To open this view, go to the **File** menu, select **New**, then click **MDX Query**.

The second is XML viewing mode, which you can get to by clicking the rightmost icon (the pencil) in the toolbar. This replaces the name/value fields with the resultant XML for each selected element. To see the entire schema, select the top-level schema entry in the element list on the left of the Schema Workbench interface. Unfortunately you won't be able to edit the XML in this view; if you want to edit it by hand, you'll have to open the schema in an XML-aware text editor.

Adding Business Groups

The available fields list in Analyzer organizes fields in folders according to the **AnalyzerBusinessGroup** annotation. To implement business groups, add these annotations to your member definitions appropriately. If no annotation is specified, then the group defaults to "Measures" for measures and the hierarchy name/caption for attributes.

Below is an example that puts Years, Quarters and Months into a "Time Periods" business group:

```
...
<Level name="Years" ... >
  <Annotations><Annotation name="AnalyzerBusinessGroup">Time Periods</Annotation></
Annotations>
</Level>
<Level name="Quarters" ... >
  <Annotations><Annotation name="AnalyzerBusinessGroup">Time Periods</Annotation></
Annotations>
</Level>
<Level name="Months" ... >
  <Annotations><Annotation name="AnalyzerBusinessGroup">Time Periods</Annotation></
Annotations>
</Level>
...
```

The AnalyzerBusinessGroup annotation is supported on the following schema elements:

- Level
- Measure
- CalculatedMember
- VirtualCubeMeasure

Adding Field Descriptions

By adding **description** attributes to your Mondrian schema elements, you can enable tooltip (mouse-over) field descriptions in Analyzer reports.

```
<Level name="Store Country" column="store_country" uniqueMembers="true"
caption="%{foodmart.dimension.store.country.caption}" description="%{foodmart.dimension.stor
>
```

This attribute can be set on the following schema elements:

- Level
- Measure
- CalculatedMember

Building a Schema and Detecting Errors

Analysis schemas are built by publishing them to the BI Server. Each schema is validated to make sure that there are no errors before it is built; if there are any, they'll be shown to you and the schema will fail to publish. If you want to see the errors marked in Schema Workbench before you publish, go to the **Options** menu and select **Require Schema**. When this option is checked, schema validation will happen as new elements are added, and any errors will show as a red **x** next to the offending element.

Restricting Access to Specific Members

You can restrict access to parts of a schema by implementing the **<HierarchyGrant>** element to define access to a hierarchy:

```
<Role name="California manager">
  <SchemaGrant access="none">
    <CubeGrant cube="Sales" access="all">
      <HierarchyGrant hierarchy="[Store]" access="custom" topLevel="[Store].[Store
Country]">
        <MemberGrant member="[Store].[USA].[CA]" access="all"/>
        <MemberGrant member="[Store].[USA].[CA].[Los Angeles]" access="none"/>
      </HierarchyGrant>
    </CubeGrant>
  </SchemaGrant>
</Role>
```

```

        <HierarchyGrant hierarchy="[Customers]" access="custom"
topLevel="[Customers].[State Province]" bottomLevel="[Customers].[City]">
            <MemberGrant member="[Customers].[USA].[CA]" access="all"/>
            <MemberGrant member="[Customers].[USA].[CA].[Los Angeles]"
access="none"/>
        </HierarchyGrant>
        <HierarchyGrant hierarchy="[Gender]" access="none"/>
    </CubeGrant>
</SchemaGrant>
</Role>

```

The **access** attribute can be **"all"**, meaning all members are visible; **"none"**, meaning the hierarchy's very existence is hidden from the user; and **"custom"**. With custom access, you can use the **topLevel** attribute to define the highest visible level (preventing users from seeing too much of the 'big picture,' such as viewing revenues rolled up to the Store or Country level); or use the **bottomLevel** attribute to define the lowest visible level (preventing users from looking at an individual customer's details); or control which sets of members a user can see by defining nested **<MemberGrant>** elements.

You can only define a **<MemberGrant>** element if its enclosing **<HierarchyGrant>** has **access="custom"**. Member grants give (or remove) access to a given member and all of its children. Here are the rules:

1. **Members inherit access from their parents.** If you deny access to California, you won't be able to see San Francisco.
2. **Grants are order-dependent.** If you grant access to USA, then deny access to Oregon, then you won't be able to see Oregon or Portland. But if you were to deny access to Oregon, then grant access to USA, you can effectively see everything.
3. **A member is visible if any of its children are visible.** Suppose you deny access to USA, then grant access to California. You will be able to see USA, and California, but none of the other states. The totals against USA will still reflect all states, however.
4. **Member grants don't override the hierarchy grant's top- and bottom-levels.** If you set `topLevel="[Store].[Store State]"`, and grant access to California, you won't be able to see USA.

Adapting Mondrian Schemas to Work with Pentaho Analyzer

The following Mondrian features are not yet functional in Pentaho Analyzer, but are scheduled to be added in the future:

- **Parent-child hierarchies** will render the entire set of members instead of showing them as parents and children.
- **Ragged hierarchies** currently throw an exception when using them in Analyzer
- Additional Mondrian features not yet available: **cube captions, drill-through, parameterization**

To adjust for these limitations and to enable some Analyzer functions to work properly, you must make the changes explained in the subsections below.

Applying Relative Date Filters

Pentaho Analyzer supports many types of relative date filters, but in order to apply them for a given level, you need to define the format string used to construct MDX members for that level. This is because each data warehouse implementation may have a different date format and set of hierarchy levels. For example, in the Steel Wheels sample data cube provided by Pentaho for evaluation and testing, the Month level uses abbreviated three-letter month names. Furthermore, the Month level sits under the Quarter level. In Steel Wheels, the format string for an MDX member from the Month level would look like this:

```
[yyyy].[ 'QTR' q].[MMM]
```

Some other common date formats:

- **[yyyy]** (Year)
- **[yyyy].[q]** (Quarter)
- **[yyyy].[q].[M]** (Month)
- **[yyyy].[q].[M].[w]** (Week)
- **[yyyy].[q].[M].[w].[yyyy-MM-dd]** (Day)

The **Day** line above also specifies a format to represent the entire date. Without this format, a simple **[d]** parameter would be difficult to put into context. For more information on date format strings, refer to the SimpleDateFormat page on the ICU Project site: <http://icu-project.org/apiref/icu4j/com/ibm/icu/text/SimpleDateFormat.html>.

To setup relative date filtering, for each level, you need to do the following:

- In your Mondrian schema file, set the **levelType** XML attribute to TimeYears, TimeMonths, TimeQuarters, TimeWeeks or TimeDate
- Define the MDX date member format as an annotation with the name **AnalyzerDateFormat**.

Here is an example from the Pentaho sample data (Steel Wheels) Time dimension:

```
<Level name="Years" levelType="TimeYears" ... >
  <Annotations><Annotation name="AnalyzerDateFormat">[yyyy]</Annotation></Annotations>
</Level>
<Level name="Quarters" levelType="TimeQuarters" ... >
  <Annotations><Annotation name="AnalyzerDateFormat">[yyyy].[ 'QTR'q]</Annotation></
Annotations>
</Level>
<Level name="Months" levelType="TimeMonths" ... >
  <Annotations><Annotation name="AnalyzerDateFormat">[yyyy].[ 'QTR'q].[MMM]</
Annotation></Annotations>
</Level>
```

Adjusting for Calculated Members and Named Sets

If your Mondrian schema defines calculated members or named sets that reference MDX members without the dimension prefix, then you must set **mondrian.olap.elements.NeedDimensionPrefix** to **false** in your **mondrian.properties** file. Under all other conditions you would want to set this property to **true** because it increases Mondrian performance, as well as the readability of the schema XML file.

Localization/Internationalization of Analysis Schemas

You can create internationalized message bundles for your analysis schemas and deploy them with the Pentaho Web application. This enables Pentaho Analyzer and JPivot to access localized schemas.

1. Edit your analysis schema and tokenize all values that you want to localize.

Typically you would create variables for all caption and description values.

```
<Schema measuresCaption="{foodmart.measures.caption}">
  <Dimension name="Store" caption="{foodmart.dimension.store.caption}"
description="{foodmart.dimension.store.description}">
  <Hierarchy hasAll="true" allMemberName="All Stores"
allMemberCaption="{foodmart.dimension.store.allmember.caption =All Stores}"
primaryKey="store_id" caption="{foodmart.hierarchy.store.country.caption}"
description="{foodmart.hierararchy.store.country.description}>
  <Table name="store"/>
  <Level name="Store Country" column="store_country"
uniqueMembers="true" caption="{foodmart.dimension.store.country.caption}"
description="{foodmart.dimension.store.country.description}"/>
```

2. Stop the BI Server.
3. Edit the **mondrian.properties** file in the `/pentaho/server/biserver-ee/pentaho-solutions/system/mondrian/` directory and add this line (or modify it if it's already there):

```
mondrian.rolap.localePropFile=com.pentaho.messages.MondrianMessages
```

4. Save and close the file.
5. Edit your `/pentaho/server/biserver-ee/pentaho-solutions/system/olap/datasources.xml` file and modify the **Provider** line to declare the default locale, include the `i18n` dynamic schema processor (this enables message token replacement), and refresh the schema when Analyzer detects that the checksum is different than the cached version:

```
Provider=mondrian; Locale=en_US; DynamicSchemaProcessor=
mondrian.i18n.LocalizingDynamicSchemaProcessor; UseContentChecksum=true; Jdbc=
jdbc:odbc:MondrianFoodMart; Catalog=/WEB-INF/FoodMart.xml
```

6. Save and close the file.

7. Create localized **MondrianMessages.properties** files in the `/WEB-INF/classes/com/pentaho/messages/` directory inside of the Pentaho WAR, and define each token you used in the analysis schema.



Note: JBoss users will have to delete the unpacked Pentaho WAR directory if it exists, then unpack the `pentaho.war` file with an archive utility, create the message bundles in the proper location, then repack it into a WAR again.

If you need further assistance in creating localized message bundles, refer to [Replacing Or Translating BI Server Message Bundles](#) on page 21.

```
foodmart.measures.caption=Measures
foodmart.dimension.store.country.caption=Store Country
foodmart.dimension.store.name.property_type.column=store_type
foodmart.dimension.store.country.member.caption=store_country
foodmart.dimension.store.name.property_type.caption=Store Type
foodmart.dimension.store.name.caption=Store Name
foodmart.dimension.store.state.caption=Store State
foodmart.dimension.store.name.property_manager.caption=Store Manager
foodmart.dimension.store.name.property_storesqft.caption=Store Sq. Ft.
foodmart.dimension.store.allmember.caption=All Stores
foodmart.dimension.store.caption=Store
foodmart.cube.sales.caption=Sales
foodmart.dimension.store.city.caption=Store City
foodmart.cube.sales.measure.unitsales=Unit Sales
```

8. Start the BI Server.

Your analysis schemas will now be localized to whatever language is currently selected in the Pentaho User Console, if a message bundle for that locale was copied to the proper directory as explained above.

Replacing Or Translating BI Server Message Bundles



Note: Stop the Pentaho application server before editing anything inside of the Pentaho WAR file.

You can internationalize the Pentaho User Console, Pentaho Analyzer, and Dashboard Designer by creating locale- and language-specific message bundles within the Pentaho Web application.

For brevity's sake, only the default Pentaho User Console files will be explained in detail. The file naming convention is identical among all message bundles in the Pentaho BI Suite. The following files are located in the `/mantle/messages/` directory:

- **mantleMessages.properties:** The default message bundle for the Pentaho User Console. In its initial condition, it is a copy of **mantleMessages_en.properties**. If you want to change the default language and dialect, copy your preferred message bundle file over this one.
- **mantleMessages_en.properties:** The English-language version of the standard message bundle. This is an identical copy of **mantleMessages.properties**.
- **mantleMessages_fr.properties:** The French-language version of the standard message bundle.
- **mantleMessages_de.properties:** The German-language version of the standard message bundle.
- **mantleMessages_supported_languages.properties:** Contains a list of localized message bundles and the native language names they correspond to; this relieves the BI Server of the burden of having to discover them on its own. A `supported_languages.properties` file should be created for every message bundle that you intend to localize.

Example of `mantleMessages_supported_languages.properties`

```
en=English
de=Deutsch
fr=Français
```

New files are created in the following format: **mantleMessages_xx_YY.properties** where **xx** represents a lowercase two-letter language code, and **YY** represents a two-letter locale code, where applicable. So, for instance, U.S. and British English could have two separate message bundles if you wanted to draw a distinction between the two dialects:


- `mantleMessages_en_US.properties`
- `mantleMessages_en_GB.properties`

The language and country codes must be in standard ISO format. You can look up both sets of codes on these pages:

- http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes


- http://www.iso.org/iso/english_country_names_and_code_elements

You can edit the default message bundle directly if you need to make surgical changes to the content inside of it. If you plan to translate it into another language, it makes more sense to copy the file and change the name appropriately, then translate it line by line. Be sure to update `supported_languages.properties` with the new country and dialect code and the native language name that it corresponds to.

 **Note:** All message bundles must be UTF-8 encoded.

Pentaho Analyzer Localization

Follow the directions below to create localized message bundles for Pentaho Analyzer.

 **Note:** This covers only the Analyzer interface, not the ROLAP data sources you use with Analyzer. For schema localization, refer to [Localization/Internationalization of Analysis Schemas](#).

1. If the BI Server is currently running, shut it down.
2. Make a copy of the **messages.properties** file in `/pentaho-solutions/system/analyzer/resources/`; name the copy according to the standard locale naming scheme defined earlier in this guide.

```
cp messages.properties messages_fr.properties
```

3. Translate the content of the new message bundle into the locale defined in its file name.
4. Edit the **messages_supported_languages.properties** file in `/pentaho-solutions/system/analyzer/resources/` and add the new locale.


```
fr=Francais
```

You now have a translated Analyzer message bundle available in your BI Server.

Setting a Default Font for PDF Exports

Before following this procedure, stop the BI Server and Pentaho Enterprise Console.

When displaying data in Analyzer, your reports will use the default browser fonts. However, the PDF export function may not have the same fonts available to it when creating a PDF from your Analyzer report, resulting in output that doesn't look the same way in PDF format as it does in the browser. The default font for PDFs is Helvetica, but you can specify any TrueType font or collection to replace it. Follow the instructions below to specify a different font for PDF exports.

 **Note:** If you have localized your schema in a language that uses a multi-byte character set (most Asian languages fit into this category), this process is required to make PDF output appear without errors.

1. Edit the **analyzer.properties** file in the `/pentaho/server/biserver-ee/pentaho-solutions/system/analyzer/` directory.
2. Uncomment the **renderer.pdf.font.path** line.

```
renderer.pdf.font.path=C:/WINDOWS/Fonts/MSGOTHIC.TTC,1
```

3. Replace the value of this line with the TrueType font or collection that you want to use as the default.

If you are specifying a collection, you must put a `,1` after the font name, as shown in the above example. This does not apply to individual fonts (TTF files).

```
renderer.pdf.font.path=/usr/share/fonts/truetype/freefont/FreeSans.ttf
```

4. Save and close the file, and start the BI Server.

Your PDF exports from Analyzer should have the font you specified.

Configuring Analysis Options

The Pentaho Analysis (Mondrian) engine is configurable through a properties file. Mondrian options allow for enhanced engine and data source performance and functionality under certain conditions. Pentaho Analyzer is also configurable, but more in terms of adjusting the functionality of the Analyzer client tool itself.

Managing Analysis Data Sources


In order to use a properly prepared data source with Pentaho Analysis, it must be established in either the Pentaho Enterprise Console as a Pentaho data connection, or in your Web application server as a JNDI data source. Whichever method you choose, you will have to know the name of the data source later on when you're asked to supply it in Pentaho Schema Workbench.

Schema Workbench can only work with one data source for each Mondrian schema; this is a structural limitation in the software that will be remedied at a later time. You can have multiple JNDI data sources or Pentaho data connections established, but only one at a time can be used with Schema Workbench.

Configuring the Mondrian Engine (mondrian.properties)

Purpose: `mondrian.properties` is a configuration file referenced by the Pentaho Analysis (Mondrian) engine to determine performance and usability settings.

Location: `/pentaho/server/biserver-ee/pentaho-solutions/system/mondrian/mondrian.properties`

 **Note:** This section is specific to maximizing Mondrian performance with Pentaho Analyzer. For all available settings, see <http://mondrian.pentaho.org/documentation/configuration.php>.

Performance

One of the key performance goals for Analyzer is to push as many operations to the database as possible. This is particularly important when querying many high-cardinality dimensions because you want Mondrian to only process the combinations that make sense and not a huge cartesian product which is sparsely populated. Pay particular attention to the following engine properties:

```
mondrian.expCache.enable=true
```

Improves the performance of summary numbers such as ranks and running sums.

```
mondrian.native.crossjoin.enable=true
```

This is an essential property which enables pushdown to the database so that Mondrian does not perform large cross joins in memory.

```
mondrian.native.filter.enable=true
```

Particularly important for pushdown of Not In and Excludes filters.

```
mondrian.native.nonempty.enable=false
```

This should be set to false to optimize Mondrian SQL generation given the MDX patterns generated by Analyzer. Analyzer uses non-empty cross joins to push down joins between high cardinality attributes to the database. If this property is set to false, there are many non-empty cross joins which would get evaluated by Mondrian instead because Mondrian pushes down the individual arguments such as **dimension.members**.

```
mondrian.rolap.maxConstraints=1000
```

Used in conjunction with the **mondrian.native.ExpandNonNative** property. This should be set to as large as the underlying DB can support.

```
mondrian.native.ExpandNonNative=true
```

Allows pushdown of even more crossjoins by materializing the crossjoin inputs into IN lists. For example, a crossjoin whose inputs are MDX functions can first have their inputs evaluated to members and then convert the crossjoin into native evaluation.

```
mondrian.olap.elements.NeedDimensionPrefix=true
```

Analyzer and JPivot always generate dimension qualified members so no need to spend time searching for members on different dimensions.

Usability

```
mondrian.result.limit=5000000
```

In the event that pushdown cannot occur, this is the largest crossjoin size that Mondrian should try to process. Anything that exceeds this will likely send the CPU for a toss and result in long server hangs. If this limit is hit, Analyzer will present the user a nice warning suggesting options to simplify the report.

```
mondrian.olap.case.sensitive=true
```

This is important for "Equals" filters because the UI will preserve the casing of filter values. If the user defines a filter on "John Doe", then the filter should only apply to "John Doe" and not "john doe"

```
mondrian.rolap.ignoreInvalidMembers=true
```

This is important for saved reports because the user may build a report with a filter on 10 sales rep and after the next ETL, one of them is gone. The report should continue to run and return just the 9 remaining sales reps.

```
mondrian.rolap.ignoreInvalidMembersDuringQuery=true
```

See `mondrian.rolap.ignoreInvalidMembers`

```
mondrian.rolap.iterationLimit=5000000
```

Similar to `mondrian.result.limit` except for controlling limits on aggregate evaluation.

```
mondrian.rolap.compareSiblingsByOrderKey=true
```

This is required for sorting members in a dimension A->Z or Z->A. This property fixes a bug in Mondrian but was added for backward compatibility.

```
mondrian.olap.NullDenominatorProducesNull=true
```

The best way to understand this property is via an example. Suppose you want to see quota attainment for your sales reps which is computed as Booked Deals / Quota. Some reps may not have quotas and so quota attainment is either infinity or null. By treating divide by null as infinity, Mondrian will return these sales reps in the report. Otherwise, if divide by null evaluates to null, then Mondrian will filter those reps out (due to the NON EMPTY).

Configuring Pentaho Analyzer (analyzer.properties)

Purpose: This property file contains low-level configuration options for the Analyzer plugin. It's sufficiently documented through inline comments, for the most part. Notes for unusual or under-documented properties are printed below.

Location: `/pentaho/server/biserver-ee/pentaho-solutions/system/analyzer/analyzer.properties`



Note: You must restart the BI Server to enact any changes you make to this file.

Refer to the sections below for more information on specific Analyzer property settings.

Adjusting Drill Link Limitations

To increase or decrease the number of records shown in drill link output, set the **report.drill.max.rows** option in **analyzer.properties** to the maximum number of rows you want to display. The default value is **10000**.

PDF Logo and Cover Page Customization

Analyzer PDF exports can be personalized with a company logo and a cover page. These image files need to be placed in the `/pentaho-solutions/system/analyzer/resources/` directory. You then need to update

analyzer.properties with the names of these images given to the keys below. If the file names are left blank, then no images are used.

- **PDF cover image:** renderer.pdf.cover.image
- **PDF logo image:** renderer.logo.image

Setting Query Limits

To set the maximum number of cells to show for a pivot table when rendered in the browser, change the value of the **renderer.browser.max.pivot.cells** property in **analyzer.properties**. The default value is **2000**

To set the maximum number of rows to export to a PDF or Excel spreadsheet, change the value of **renderer.export.max.rows**. The default is **10000**.

Divide by Zero Display

The **renderer.infinite.nan.value.display** property in **analyzer.properties** determines what is shown to users when a "divide by zero" error occurs in an Analyzer report. You can put any string (no quotes necessary) here; Excel uses **DIV/0**. The default value is **N/A**.

Defining Global Default Chart Options in Analyzer

The **analyzer.properties** file can be modified to change many of the default chart settings in Analyzer. The table below explains the options that are currently available.

Analyzer Chart Properties

The **analyzer.properties** settings below pertain to global default chart options in Analyzer.

Option	Definition
chart.options.legendPosition	Chart legend position. Possible values are: TOP, RIGHT, BOTTOM, LEFT
chart.options.showLegend	Show legend by default. Possible values are: true, false
chart.options.lineWidth	Line chart width in pixels
chart.options.lineShape	Line chart data point shape. Possible values are: NONE, CIRCLE, DIAMOND, SQUARE, TRIANGLE
chart.options.maxValues	Chart maximum domain values
chart.options.backgroundColor	Chart background color in hexadecimal format
chart.options.labelColor	Chart title and axis label colors in hexadecimal format
chart.options.labelSize	Chart axis label font size in points
chart.options.backgroundFill	Chart background fill type. Possible values are: NONE, SOLID, GRADIENT
chart.options.maxChartsPerRow	Number of charts per row for multi-charts
chart.options.backgroundColorEnd	Chart background end color (only applicable for GRADIENT) in hexadecimal format
chart.options.labelStyle	Chart label font style - NORMAL, ITALIC, BOLD
chart.options.legendBackgroundColor	Chart legend background color in hexadecimal format
chart.options.legendSize	Legend font size in points
chart.options.legendColor	Legend font color in hexadecimal format
chart.options.legendStyle	Chart legend font style. Possible values are: NORMAL, ITALIC, BOLD
chart.series.colors	Sets the default series color palette. See the section below for details.


Setting the default chart palette

You can set the default series color palette in Analyzer through the **chart.series.colors** element in `analyzer.properties`. The values are a comma-separated list of hexadecimal color values for the series members in a chart.

```
# The below setting is the standard Analyzer series color palette  
chart.series.colors=#5a84ce,#a5bd5a,#8c63ad,#5aadd6,#ce6a5a,#cea45a,#5acea4,#27519b,#ce5a84,#
```

Mondrian Cache Control

This section contains instructions for configuring and controlling the cache infrastructure that the Pentaho Analysis engine uses for ROLAP data. This information is useful for properly updating your ROLAP cubes when your data warehouse is refreshed, and for performance-tuning.


 **Restriction:** Most of the advanced cache features explained in this section are for Enterprise Edition deployments only. Within that, most of the Enterprise Edition features of the Analysis engine are only beneficial to large, multi-node ROLAP deployments that are performing poorly.

By default, the Analysis engine is configured to use a distributed cache through JBoss **Infinispan**. Infinispan supports a wide variety of sub-configurations and can be adapted to cache in memory, to the disk, to a relational database, or (the default setting) to a distributed cache cluster.


The Infinispan distributed cache is a highly scalable solution that distributes cached data across a self-managed cluster of Mondrian instances. Every Mondrian instance running the Analysis Enterprise Edition plugin on a local network will automatically discover each other using UDP multicast. An arbitrary number of segment data copies are stored across all available nodes. The total size of the cache will be the sum of all of the nodes' capacities, divided by the number of copies to maintain. This is all fully configurable; options are explained later in this section.

Other supported segment cache configurations include, but are not limited to:

- **Memcached**, which uses an established (extant) Memcached infrastructure to cache and share the segment data among Mondrian peers.
- **Pentaho Platform Delegating Cache**, which relies on the Pentaho BI Server to delegate segment data storage to the BI Server's native caching capabilities, thus leveraging the existing caching configuration. Some people may prefer this configuration because it keeps the BI Server and Analysis engine manageable as a single entity.

 **Note:** The Pentaho Platform Delegating Cache is not yet feature-complete. It will be available in a future BI Suite release, but it is not yet ready for production use.

Segment Cache Architecture

 **Restriction:** The segment cache features explained in this section are for very large ROLAP deployments, and require a Pentaho Analysis Enterprise Edition license.

How the Analysis Engine Uses Memory

Each Mondrian segment cache node, regardless of which configuration it uses, loads the segments required to answer a given query into system memory. This cache space is called the **query cache**, and it is composed of hard Java references to the segment objects. Each individual node must have enough memory space available to answer any given query. This might seem like a big limitation, but Mondrian uses deeply optimized data structures which usually take no more than a few megabytes, even for queries returning thousands of rows.

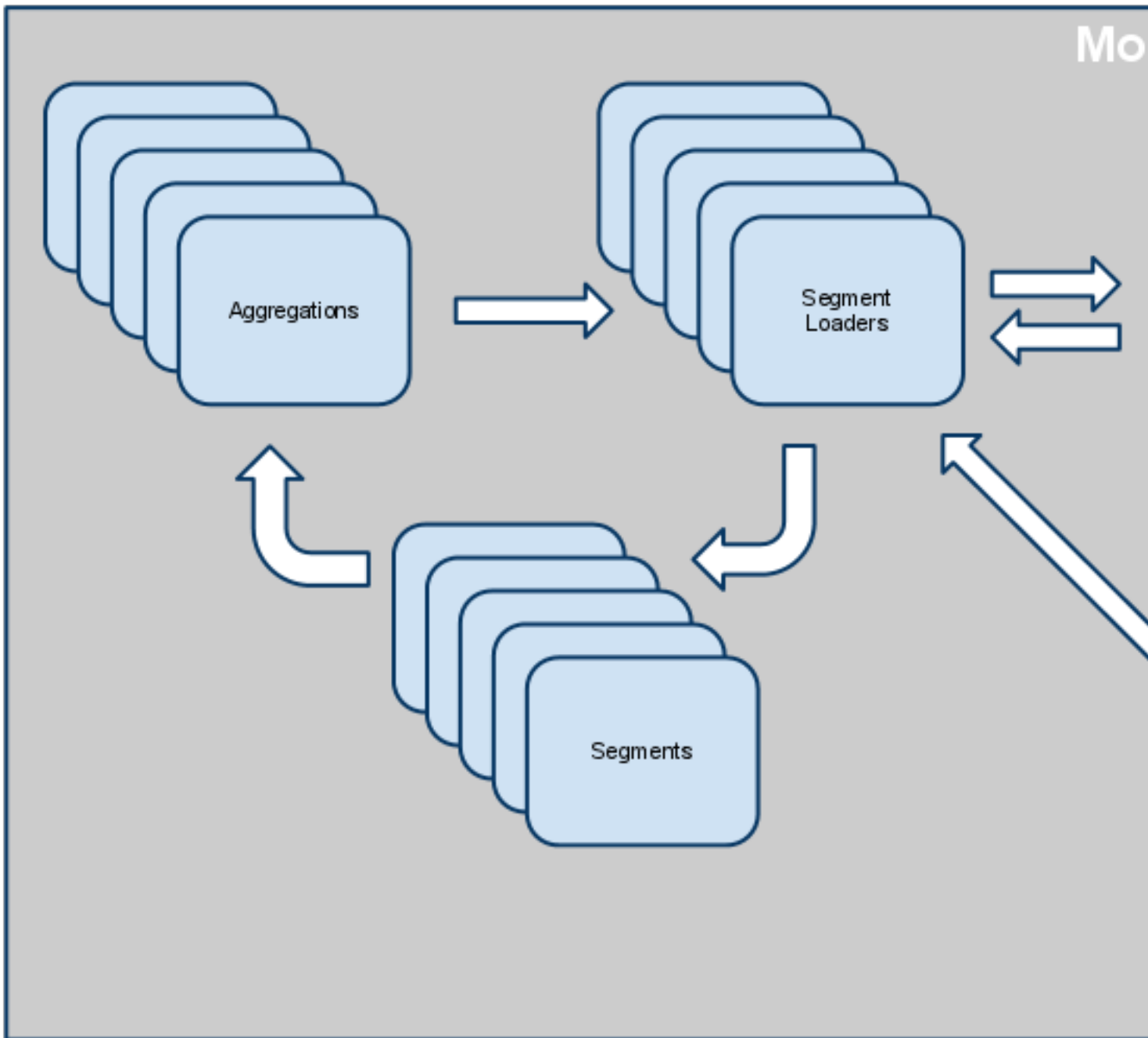
Once the query finishes, Mondrian will usually try to keep the data locally, using a weak reference to the segment data object. A weak reference is a special type of Java object reference which doesn't force the JVM to keep this object in memory. As the Mondrian node keeps answering queries, the JVM might decide to free up that space for something more important, like answering a particularly big query. This cache is referred to as the **local cache**.

The local cache can be switched on or off by editing the Pentaho Analysis EE configuration file and modifying the value (set it to **true** or **false**) of the **DISABLE_LOCAL_SEGMENT_CACHE** property. Setting this property will not affect the query cache.

This is the order in which Mondrian will try to obtain data for a required segment once a query is received:

1. The node will parse the query and figure out which segments it must load to answer that particular query
2. It checks into the local cache, if enabled.
3. If the data could not be loaded from the local cache, it checks into the external segment cache, provided by the Pentaho Analysis plugin, and it places a copy inside the query cache.
4. If the data is not available from the external cache, it loads the data from SQL and places it into the query cache.
5. If the data was loaded from SQL, it places a copy in the query cache and it sends it to the external cache to be immediately shared with the other Mondrian nodes.

6. The node can now answer the query.
7. Once the query is answered, Mondrian will release the data from the query cache.
8. If the local cache is enabled, a weak reference to the data is kept there.



Cache Control and Propagation

All cache control operations are performed through Mondrian's CacheControl API. It is documented in the Mondrian project documentation. The CacheControl API allows you to modify the contents of the cache of a particular node. It controls both the data cache, but also the cache of members which form your OLAP schema. For the purposes of this document, we will only cover operations related to the data cache, or segment cache, and how it relates to the external segment cache.

When flushing a segment region on a node, that node will propagate the change to the external cache, using the SegmentCache SPI. If the nodes are not using the local cache space, you can safely assume that the data will be fetched again through SQL by the next node which picks up a query requiring that segment. Once the data is loaded from SQL, it will again be stored in the external segment cache.

The Mondrian team does not recommend using the local cache space when using the external cache. This is why we have disabled it by default when you deploy the Pentaho Analysis plugin.


Lets assumes that you have turned on the local segment cache. Using the local cache space on a node can improve performance with increased data locality, but it also means that all the nodes have to be notified of that that change. At the moment, Mondrian nodes don't propagate the cache control operations between the members of a cluster. If you deploy a cluster of Mondrian nodes and you do not propagate the change manually across the nodes, you will have a situation where some nodes will answer queries with stale data.

Cache Configuration Files


The following files contain configuration settings for Pentaho Analysis cache frameworks. All of them are in the same directory inside of the deployed pentaho.war: `/WEB-INF/classes/` .

- **pentaho-analysis-config.xml** Defines the global behavior of the Pentaho Analysis Enterprise Edition plugin. Settings in this file enable you to define which segment cache configuration to use, and to turn off the segment cache altogether.
- **infinispan-config.xml** The InfinispanSegmentCache settings file. It configures the Infinispan system.
- **jgroups-udp.xml** Configures the cluster backing the Infinispan cache. It defines how the nodes find each other and how they communicate. By default, Pentaho uses UDP and multicast discovery, which enables you to run many instances on a single machine or many instances on many machines. (There are examples of other communication setups included in the JAR archive.) This file is referenced by infinispan as specified in the infinispan-config.xml configuration file.
- **memcached-config.xml** Configures the Memcached-based segment cache. It is not used by default. To enable it, modify **SEGMENT_CACHE_IMPL** in pentaho-analysis-config.xml.

Modifying the JGroups Configuration

 **Restriction:** The segment cache features explained in this section are for very large ROLAP deployments, and require a Pentaho Analysis Enterprise Edition license.

The default Infinispan configuration uses JGroups to distribute the cache across all Mondrian instances it finds on the local network. If you want to modify how those communications are done, you must edit the JGroups configuration file.


 **Note:** Fine-grained JGroups configuration is covered in the JGroups documentation; you should read through it before making changes.

Each node might require a different configuration, so although the default configuration is highly portable, it might not work for you.

If you are deploying this plugin on Amazon EC2, JGroups has a special configuration file that you copied to your `/WEB-INF/classes/` directory when you installed the Analysis Enterprise Edition package. Additionally, default JGroups configuration files are inside of the JAR archive. To switch implementations, edit **infinispan-config.xml** and make the modification appropriate to your communication method:

Comm. type	Config entry
UDP communication	<code><property name="configurationFile" value="jgroups-udp.xml" /></code>
TCP communication	<code><property name="configurationFile" value="jgroups-tcp.xml" /></code>
Amazon EC2	<code><property name="configurationFile" value="jgroups-ec2.xml" /></code>

Switching to Another Cache Framework

 **Restriction:** The segment cache features explained in this section are for very large ROLAP deployments, and require a Pentaho Analysis Enterprise Edition license.

Pentaho Analysis Enterprise Edition ships with Infinispan cache support enabled by default. Instructions are provided below for switching to the Pentaho Platform Delegating Cache or Memcached. There is also a brief overview of how to create a Java class to implement your own cache system.

Switching to Memcached

In order to complete this procedure, you must have your own pre-configured Memcached instance. You should have also installed the Analysis Enterprise Edition package to your BI Server or standalone Mondrian engine.

If you already use the Memcached cache framework in your organization and would like to hook it up to the Pentaho Analysis ROLAP engine, follow the directions below to switch from the default Infinispan cache framework.

1. If the BI Server or standalone Mondrian engine are running, shut them down now.
2. If you performed a default install of the Pentaho Analysis Enterprise Edition package, then you should have all of the required JARs installed to the BI or Mondrian server. If you aren't sure, verify now that the following JARs are present in the `/WEB-INF/lib/` directory inside of your deployed `pentaho.war` or Mondrian engine:

- pentaho-analysis-ee
- commons-lang
- commons-io
- commons-codec
- pentaho-ee-dsc-core
- memcached

3. Edit the `pentaho-analysis-config.xml` in the `/WEB-INF/classes/` directory inside the deployed `pentaho.war` or Mondrian engine, and change the value of `SEGMENT_CACHE_IMPL` to match the class name referenced below:

```
<entry key="SEGMENT_CACHE_IMPL">com.pentaho.analysis.segmentcache.impl.memcached.MemcachedSegmentCache</entry>
```

4. Edit the `memcached-config.xml` in the `/WEB-INF/classes/` directory inside the deployed `pentaho.war` or Mondrian engine, and change the values of `SALT`, `SERVERS`, and `WEIGHT` to match your preference:

Refer to [Memcached Configuration Options](#) on page 30 for more information on these settings.

```
<entry key="SALT">YOUR SECRET SALT VALUE HERE</entry>


<entry key="SERVERS">192.168.0.1:1642,192.168.0.2:1642</entry>

<entry key="WEIGHTS">1,1</entry>
```

Your Pentaho Analysis Enterprise Edition instance is now configured to use Memcached for ROLAP segment caching.

Memcached Configuration Options

These properties control Memcached settings, and are set in the `memcached-config.xml` file in the `/WEB-INF/classes/` directory inside of your deployed `pentaho.war` or Mondrian engine.

 **Note:** This is not a comprehensive list of the potential Memcached settings; the options explained below are the ones most critical to Memcached configuration for Pentaho Analysis. You must research further Memcached options on your own.

Property	Purpose
SERVERS	A comma-separated list of servers and port numbers representing the Memcached nodes usable by the plugin.
WEIGHTS	A comma-separated list of numbers representing the relative caching capacity of the servers defined in the SERVERS property. There must be exactly as many values of WEIGHTS as there are values of SERVERS. As an example, if the first server has a capacity of 128 megabytes, and the second has a capacity of 256 megabytes, the correct values for the WEIGHTS property should be "1,2", indicating that the first server has a relative size of half of the second one.
SALT	A secret key prefix to be used when saving and loading segment data from the Memcached nodes. This property must be the same for all Mondrian nodes that share their

Property	Purpose
	caches. If the SALT value is different from one node to the next, the nodes will not be able to share their cache data.

Switching to Pentaho Platform Delegating Cache

In order to complete this procedure, you must have installed the Analysis Enterprise Edition package to your BI Server. If you would like to share the BI Server solution cache with the Pentaho Analysis segment cache, follow the directions below.

This cache system is still experimental and not fully implemented; it is not recommended for production use. Therefore, no public documentation is available at this time.

Using a Custom SegmentCache SPI

If you want to develop your own implementation of the SegmentCache SPI, you'll have to follow this basic plan:

1. Create a Java class that implements **mondrian.spi.SegmentCache**
2. Compile your class and make it available in Mondrian's classpath
3. Edit **mondrian.properties** and set **mondrian.rolap.SegmentCache** to your class name
4. Start the BI Server or Mondrian engine

This is only a high-level overview. If you need more specific advice, contact your Pentaho support representative and inquire about developer assistance.

Clearing the Mondrian Cache

There is a default action sequence in the BI Server that will clear the Mondrian cache, which will force the cache to rebuild when a ROLAP schema is next accessed by the BI Server.

The cache-clearing action sequence is **clear_mondrian_schema_cache.xaction**, and you can find it in the **admin** solution directory. This action sequence can be run directly from a URL by making the **admin** solution directory visible and then running the action sequence from the solution browser in the Pentaho User Console, from the Pentaho User Console by selecting the **Mondrian Schema Cache** entry in the **Refresh** part of the **Tools** menu, or by clicking the **Mondrian Cache** button in the **Administration** section of the Pentaho Enterprise Console:

```
http://localhost:8080/admin/clear_mondrian_schema_cache.xaction
```

Priming the Cache

This procedure is not yet complete.

Task step.

Mondrian Role Mapping in the BI Server

The purpose of the role mapper feature is to map roles defined in a Mondrian schema to user roles in the Pentaho BI Server. You can edit the mapper configuration as needed to meet the needs of your organization.

The role mapper exists and is enabled in the Pentaho BI Server by adding a new bean to the file, `pentahoObjects.spring.xml`, located at `...\pentaho-solutions\system`. There are currently three different mapper implementations to choose from, and each is described below. The `pentahoObjects.spring.xml` file also has example configurations for each of these mappers commented out. The end result of each mapper is to pass the granted roles for users who are logged on to the BI server to Mondrian.

In instances in which a role is passed to Mondrian through the definition in an action sequence, (this is the original method; see [Using Security Information In Action Sequences](#)), the role-related data in the action sequence overrides any and all mappers. If you do not use a mapper, none of the roles defined in Mondrian are honored and the schema is left wide open.

The Mondrian-One-To-One-UserRoleMapper

This Mondrian user/role mapper assumes that roles in the BI platform also exist in Mondrian; for example if you have a role called "CTO" in Mondrian and in the BI platform. The **Mondrian-One-To-One-UserRoleMapper** maps each role in the platform to roles defined in the Mondrian schema by name.

```
<bean id="Mondrian-UserRoleMapper"
name="Mondrian-One-To-One-UserRoleMapper"
class="org.pentaho.platform.plugin.action.mondrian.mapper.
MondrianOneToOneUserRoleListMapper"
scope="singleton" />
```

The Mondrian-SampleLookupMap-UserRoleMapper

This sample mapper assumes that a translator is needed (in the form of a map) to map a BI platform role to a Mondrian role. So the map provides the lookups needed for each role. The lookups take the form of Key = BI platform role, Value = mondrian role. In the sample below, there is a "ceo" role in the BI platform which maps to the "California manager" role in Mondrian.

```
<bean id="Mondrian-UserRoleMapper"
name="Mondrian-SampleLookupMap-UserRoleMapper"
class="org.pentaho.platform.plugin.action.mondrian.mapper.
MondrianLookupMapUserRoleListMapper"
scope="singleton">
<property name="lookupMap">
<map>
<entry key="ceo" value="California manager" />
<entry key="cto" value="M_CTO" />
<entry key="dev" value="M_DEV" />
</map>
</property>
</bean>
```

The Mondrian-SampleUserSession-UserRoleMapper

This sample mapper assumes that all users have Mondrian roles in their session under a named session variable.

```
<bean id="Mondrian-UserRoleMapper"
name="Mondrian-SampleUserSession-UserRoleMapper"
class="org.pentaho.platform.plugin.action.mondrian.mapper.
MondrianUserSessionUserRoleListMapper"
scope="singleton">
<property name="sessionProperty" value="MondrianUserRoles" />
</bean>
```


Visualizing Your Data


Pentaho provides two end-user client tools that are designed specifically for traversing an analysis schema: JPivot, an open source analysis viewer that is useful for creating MDX queries; and Pentaho Analyzer, a highly interactive and easy to use analysis slicer-dicer.

Once you have JPivot or Pentaho Analyzer content, you can use Dashboard Designer to create a dashboard that includes your analysis view or Analyzer report plus any other content in a variety of configurations and styles.

It is also possible to use your analysis schema as a data source in Pentaho Report Designer, though you will have to create your own MDX query to refine the data for reporting.

Introduction to the Multidimensional Expression Language (MDX)

The Multidimensional Expression Language (MDX) is an OLAP database query and calculation language similar to SQL. It is the method by which a dataset is retrieved from a ROLAP database. While you do not necessarily have to know MDX in order to use Pentaho's client tools, you should be somewhat familiar with it for the purpose testing your Mondrian schema.

 **Note:** The JPivot analysis viewer built into the Pentaho User Console is an excellent way to quickly generate a valid MDX query without having to write any of it by hand.

If you are already familiar with SQL, then much of the MDX syntax will look familiar, and the rest should be relatively easy to learn.

There are six MDX data types:

1. Dimension or hierarchy
2. Level
3. Member
4. Tuple
5. Scalar
6. Set


Below is an example of a very simple MDX query:

```
SELECT
  { [Measures].[Salesfact] } ON COLUMNS,
  { [Date].[2004], [Date].[2005] } ON ROWS
FROM Sales
```

Further Reading

MDX was initially developed by Microsoft for its SQL Server analysis products, though it has since become an independent standard. It's been around long enough now that there are many MDX tutorials and references, most notably:

- <http://msdn2.microsoft.com/en-us/library/ms145506.aspx>
- http://en.wikipedia.org/wiki/Multidimensional_Expressions
- http://www.anselmiconsulting.com/webtools/WTContent/ACDR/Tutorials/MDXReportTutorial/MDXTutorial/mdx_tutorial.htm

 **Note:** MDX implementations vary, and many MDX documentation resources are specific to certain niche products or standards. Not all MDX functions and extensions are supported in Pentaho Analysis.

Creating Analyzer Reports and Analysis Views

Pentaho offers two primary slicer-dicer tools: **Pentaho Analyzer** and **JPivot**.

Pentaho Analyzer is a powerful and extremely easy-to-use visualization tool that is delivered as a plugin to the Pentaho User Console. This is by far the preferred client tool for generating analysis content.

JPivot is an open source slicer-dicer with a simple, lightweight interface and good native Mondrian support. It's not nearly as user-friendly as Analyzer is, but it is still useful for generating good MDX queries.

The sections below explain how to use these tools on a basic level.

Creating a New Analyzer Report

Pentaho Analyzer is an interactive analysis tool that provides you with a rich drag-and-drop user interface, which makes it easy for you to create reports based on your exploration of your data. You can also display Pentaho Analyzer reports in a dashboard. Unlike standard reports which tend to be static or minimally interactive after they are created, Pentaho Analyzer reports allow you to explore your data dynamically and to drill down into the data to discover previously hidden details. Pentaho Analyzer presents data multi-dimensionally and lets you select what dimensions and measures you want to explore. Use the Analyzer to drill, slice, dice, pivot, filter, chart data and to create calculated fields. Users of Analyzer reports can query the data in a database without having to understand how the database is structured.

Note: The field view in the sample report below is organized **by Category** (hierarchies and levels within a dimension). You can organize the field view in your Analyzer report alphabetically (**A-Z**), by **Number** (measures) - **Text** (names, categories, etc.) - **Time** (months, quarters, and years), and by **Schema**, which displays the dimensions as defined in the Mondrian schema.


Territory	Line	Years					
		2003			2004		
		Sales	Quantity	Unit Sales	Sales	Quantity	Unit Sales
APAC	Classic Cars	\$115,011	1,052	\$109	\$199,372	1,785	\$112
	Vintage Cars	\$111,639	1,243	\$90	\$147,212	1,587	\$93
	Motorcycles	\$60,789	654	\$93	\$63,159	540	\$117
	Trucks and Buses	\$11,298	91	\$124	\$80,634	801	\$103
	Planes	\$42,663	456	\$94	\$67,681	723	\$94
APAC Total		\$341,400	3,496	\$98	\$558,057	5,436	\$103
APAC Sum		\$341,400	3,496	\$98	\$558,057	5,436	\$103
EMEA	Classic Cars	\$691,273	5,853	\$118	\$1,015,790	8,024	\$113
	Vintage Cars	\$263,695	3,094	\$85	\$504,062	5,472	\$92
	Motorcycles	\$141,836	1,428	\$99	\$204,042	2,177	\$94
	Trucks and Buses	\$228,699	2,261	\$101	\$185,421	1,558	\$119
	Planes	\$154,519	1,723	\$90	\$209,128	2,326	\$90
EMEA Total		\$1,480,021	14,359	\$103	\$2,118,443	20,509	\$103
EMEA Sum		\$1,480,021	14,359	\$103	\$2,118,443	20,509	\$103
Japan	Classic Cars	\$120,696	898	\$134	\$42,071	307	\$137
	Planes	\$60,556	677	\$89	\$49,177	547	\$90
	Trucks and Buses	\$44,498	415	\$107	\$13,349	102	\$131
	Motorcycles	\$16,485	205	\$80	\$31,959	380	\$84
	Vintage Cars	\$22,898	308	\$74	\$21,470	229	\$94

In this exercise, you will be creating a report that displays the actual versus budgeted expenses by region for each department in the fictitious Steel Wheels company (SampleData).


Follow the instructions below to start creating the Analyzer report.

Note: If you do not have access to Sample Data, adapt the instructions to your own data source.

1. In the Pentaho User Console menubar, go to **File -> New** and select **Analyzer Report**. Pentaho Analyzer opens.
2. Select your **Schema** and **Cube** from the list. For the purposes of this exercise, select **SampleData:Quadrant Analysis**.

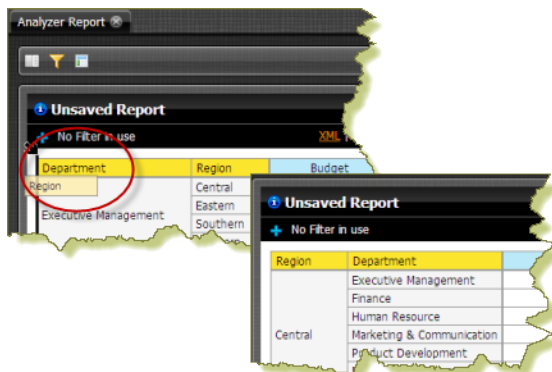
 **Note:** The list of available schemas and cubes are provided by your administrator. A **schema** is the structure of the relational database and includes tables, fields, views, and more. A **cube** is a data structure that allows information in a database to be analyzed quickly and from multiple perspectives.




3. In the list of fields (on the right), click and drag **Department** to the **Analyzer** workspace in the left pane. The **Department** column appears in the workspace.
4. In the list of fields, click and drag **Budget** to the **Analyzer** workspace in the left pane. The **Budget** column appears in the workspace.
5. Right-click the **Budget** column to display the **Edit Column** dialog box. Select **Column Name and Format** -> **Currency (\$)** from the **Format** list so that your values display as dollar amounts. Click **OK**.
6. In the list of fields, click and drag **Actual** to the **Analyzer** workspace in the left pane. The **Actual** column appears in the workspace.
7. Right-click the **Actual** column to display the **Edit Column** dialog box. Select **Column Name and Format** -> **Currency (\$)** from the **Format** list so that your values display as dollar amounts. Click **OK**.
8. Click and drag the **Region** column to the report. (Read the note below.)

 **Note:** Notice that the **Region** column appears in the workspace; however, you want this column to be the first column in the report.

9. In the workspace, click and drag the **Region** column to the left of the **Department** column.

Pentaho Analyzer is designed to provide you with great flexibility when designing the visual structure of your report.



10. At this point you have a functioning report and you can view your data in chart form. Click  (Switch to Chart Format) to examine your report data in a chart format. The default display is a bar chart but if you click  (Choose Chart Type) you can select a different chart type to display your data.
11. Save your report before continuing the exercise. In the Pentaho User Console, click  (Save As). When the **Save As** dialog box appears, save your report as **Regional Expense Report** under /Steel Wheels/Analysis and click **OK**.

Managing Charts in Pentaho Analyzer

The charting features in Pentaho Analyzer give you the flexibility to control the look and feel of the charts in your Analyzer reports.


Setting Chart Options

The **Set Chart Options** allow you to define the chart display (colors, legend, number of charts per row, bullet styles, and more). To access the Chart Options dialog box, click **More** in the upper right corner of the Pentaho Analyzer toolbar and choose, **Set Chart Options**.



Chart options are defined below:

General

Option	Description
Background	<p>Fill type defines the background color of the chart and the specific fill colors used. A Gradient will result in a gradual color transition in the fill color. Choose the color you want from the available color pickers (Fill Color, End Color) You can select not to have a fill color (None), or select a single background color (Solid).</p>
Labels	<p>Allows you to select a font type, size, formatting, and color of the labels in your chart.</p>
Axis	<p>Allows you to adjust the axis range. When set to Auto Range, the chart will scale based on the data. If you have a time series chart, you may want to set the range manually so that figures, (not close to the 0 axis), display correctly; (see image below). Select a Scale option to manage instances where numeric values in a chart are so long that they affect the display.</p> 

Legend

Option	Description
Show Legend	Allows you to enable to disable the legend display
Position	Allows you to select the placement (Top, Right, Bottom, Left) of the legend on the chart
Background Color	Allows you to select a background color for the legend
Font	Allows you to set the font type, size, format, and color associated with your legend

Other

Option	Description
Line Charts	Allows you to set the Bullet Style for points on a line chart and assign Line Width
Multi-Charts	Allows you to define the number of charts that display when using the Multi-Charts feature
Domain Limit	Allows you to limit the number of plot values that display in a chart.

Displaying Multi-Charts in Analyzer

The Analyzer multi-chart feature allows you to display multiple charts in a single Analyzer report. For example, you can use this feature when you want to display the same data over different years. The multi-chart feature prevents you from having to create separate filters and reports.



1. Create a New Analyzer report.



Important: You must have Auto-Refresh Mode disabled (see [Disabling Auto Refresh Mode](#)) to use the Multi-Chart feature.

- 2.

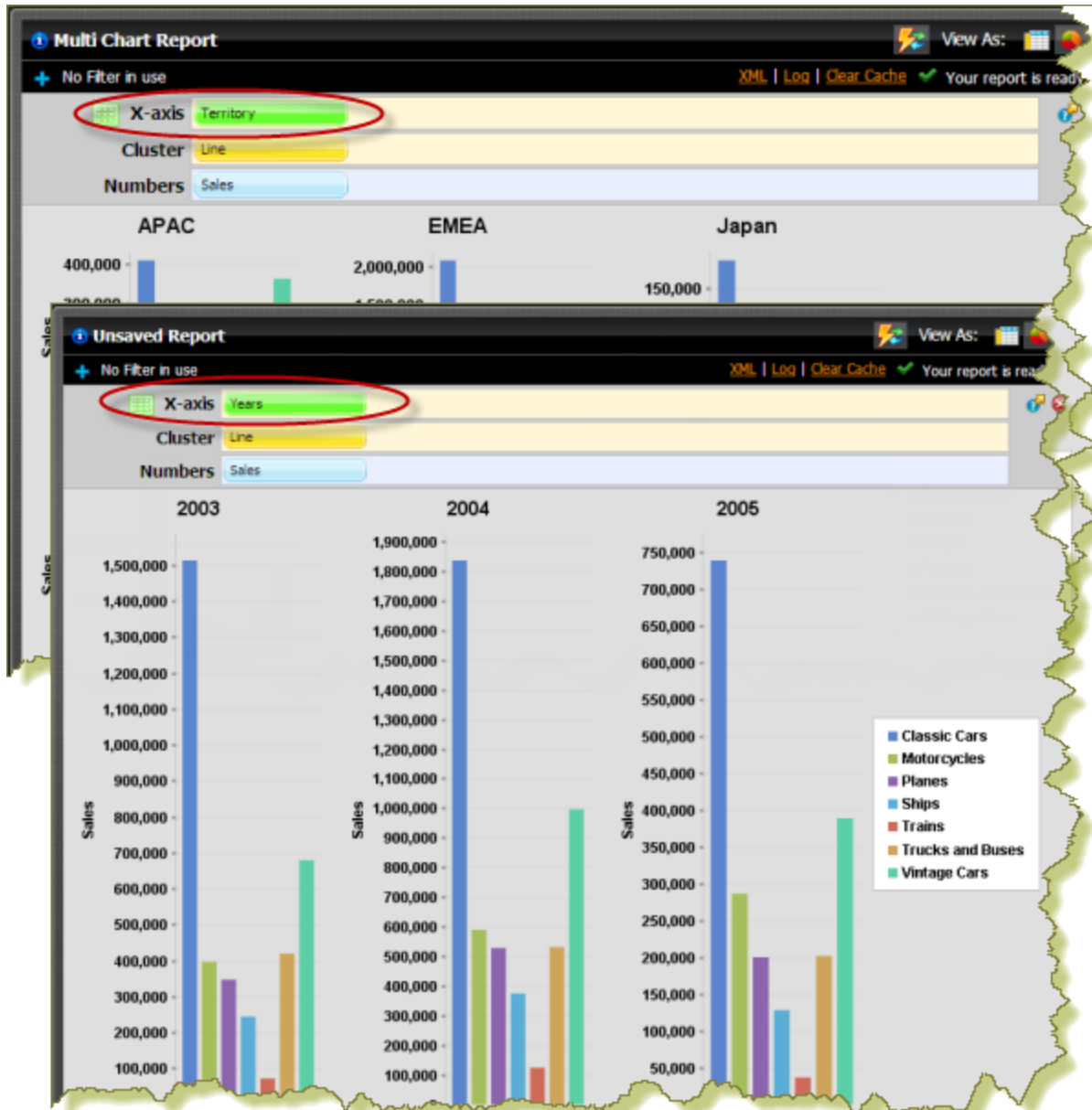


Click  to change to chart format mode.

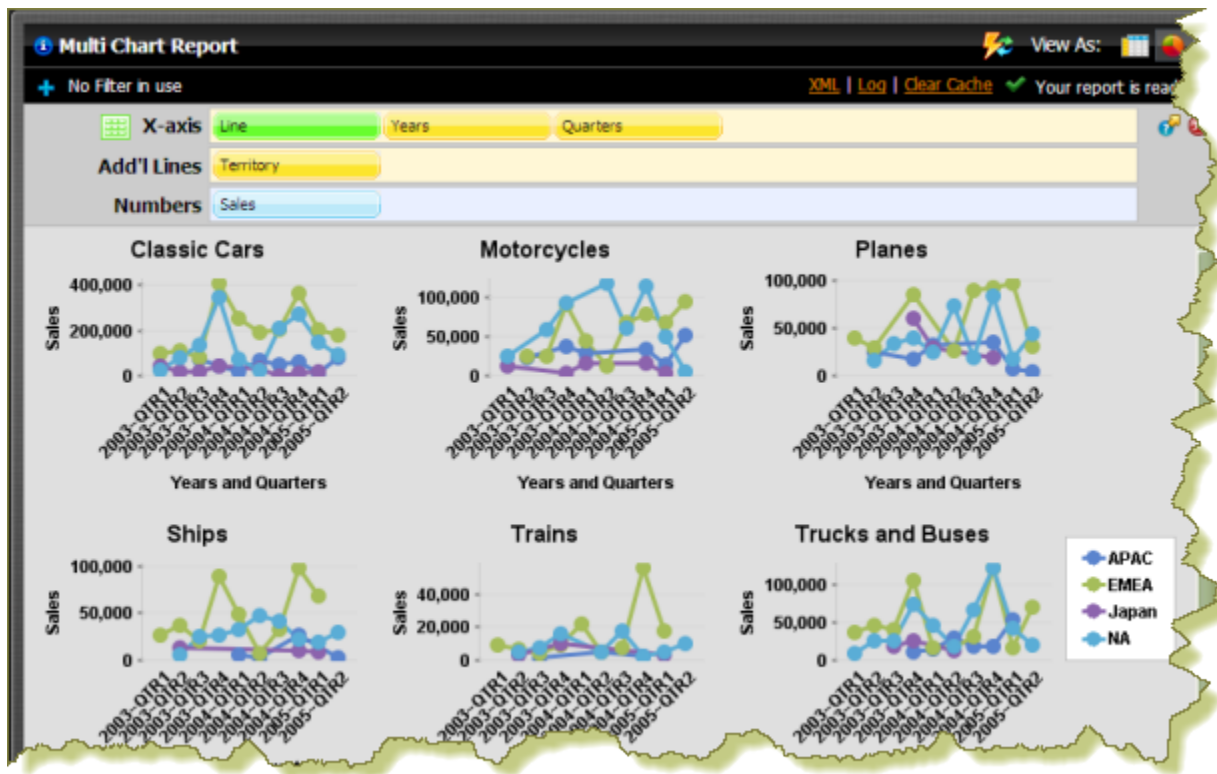
3. Move the appropriate fields into the field layout section.
4. Click **Enable Multi-Charts** in the upper left corner of the field layout section.



The field that you defined as the **X-axis** will become the data to be sliced into multiple charts. In the first example, **Territory** is defined as the X-axis and the report displays data by Territory. In the second example, the **Years** is defined as the X-axis and the report displays data by Years.



You can change the chart type, rearrange the fields, or add new fields as needed.



5. Save your report.

Drilling in Charts

The Pentaho Analyzer drill down feature in charts, allows you to click on a single bar in a bar chart, a wedge in a pie chart, or a line in a line chart to drill down for more details. Drill downs display in a separate chart.

Consider the example below. When hovering over a bar in the chart, a tooltip describes the data associated with that particular bar; in this example, **Vintage Car Sales and EMEA; click to show Vendor**. Once you click the bar, a new chart appears that displays the vendors associated with the product line (Classic Cars, Motorcycles, Planes, and so on) for the territory, EMEA. The drill down path follows the levels in the hierarchy. You can continue drilling down until there are no more levels of data to display. Drill down also works with multi-charts.




Adding Query Parameters to Analyzer Reports

You must be logged into the Pentaho User Console, and have an Analyzer report open for editing in order to continue. Follow the below procedure to parameterize your MDX query in Analyzer.


1. Right-click on the dimension member you want to create a parameter for, and select **Filter** from the context menu.
2. Select the level you want to set as the default parameter value, then click the right arrow to move it to the list on the right.
3. Click the checkbox at the bottom of the window, then type in a name for the parameter in the **Parameter Name** field.
4. Click **OK**.


Your parameter is now created as a filter in Pentaho Analyzer. Whenever this Analyzer report is run, users will have a selection of columns to filter by.

 **Note:** If you create a dashboard with this Analyzer report, then this filter can be used by Dashboard Designer as a parameter as well.

Configuring Drill-Down Links

To create reports based on specific number value data, you can implement drill-down links in Pentaho Analyzer. This will turn all non-calculated number fields into links which, when clicked, bring up a configurable data grid that enables you to quickly view more details for that data point without having to reconfigure your report. Follow the directions below to turn on drill-linking.

 **Note:** Calculated members are unavailable for drill-down at this time.

 **Note:** Drill-down links will not work in reports that have filters that are not being used. If you have any filters in an Analyzer report, they must be used in the report in order to view drill-down links.

1. Open the Analyzer report that you want to add drill links to.
2. Go to the **More** menu in the upper right corner of the report, and select **Set Report Options** from it.
3. Select the **Show drillthrough links on Number cells** checkbox, then click **OK**.
The number fields in your report will turn into links.
4. Click on a drill-down link to see a data grid that shows all of the available details for that value.
5. To add or remove columns from the grid, click on the down arrow on the right side of any column and select the **Columns** sub-menu. From there you can select which columns you want to appear. You can also sort by ascending or descending values for any column through this menu.

You now have drill-down links for numeric, non-calculated members.

Filtering by Member Properties

If a dimension has a number in parenthesis next to it in the field list, that means that it has **member properties** associated with it. To constrain a dimension by controlling its member properties, follow the instructions below.

1. Open the field layout by clicking **Show Field Layout** button above the grid.
Depending on how you have them oriented, you will see your selected dimensions in either the **Row Labels** or **Col Headers** fields.
2. Right-click on a dimension in the row label or column header fields, then select **Show Properties** from the context menu.
A sub-menu with all available member properties will appear.
3. Check or un-check the member property boxes to add or remove them from the report, then click **OK**.

The analyzer report will refresh and show the filter choices you've selected.

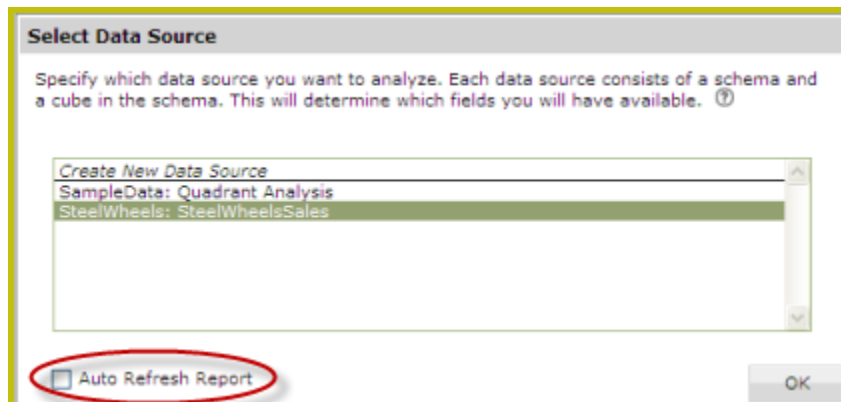
Disabling Auto Refresh Mode

When you disable the **Auto Refresh Report** feature in Analyzer you can design your report layout first, including calculations and filtering, without querying the database until you are done. Once the report layout is complete, you can re-enable Auto Refresh Report. Data retrieval will occur once and your report will display the requested data.


Follow the instructions below to disable auto refresh and build a report.

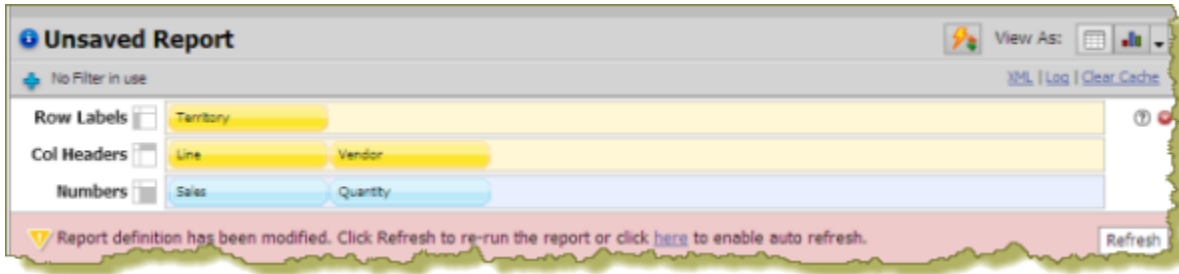
1. In the Pentaho User Console menubar, go to **File -> New** and select **Analyzer Report**.
Pentaho Analyzer opens.
2. Select your Data Source, (schema and cube), from the list.
3. Disable **Auto Refresh Report**.

 **Note:** Auto Refresh Report is enabled by default.

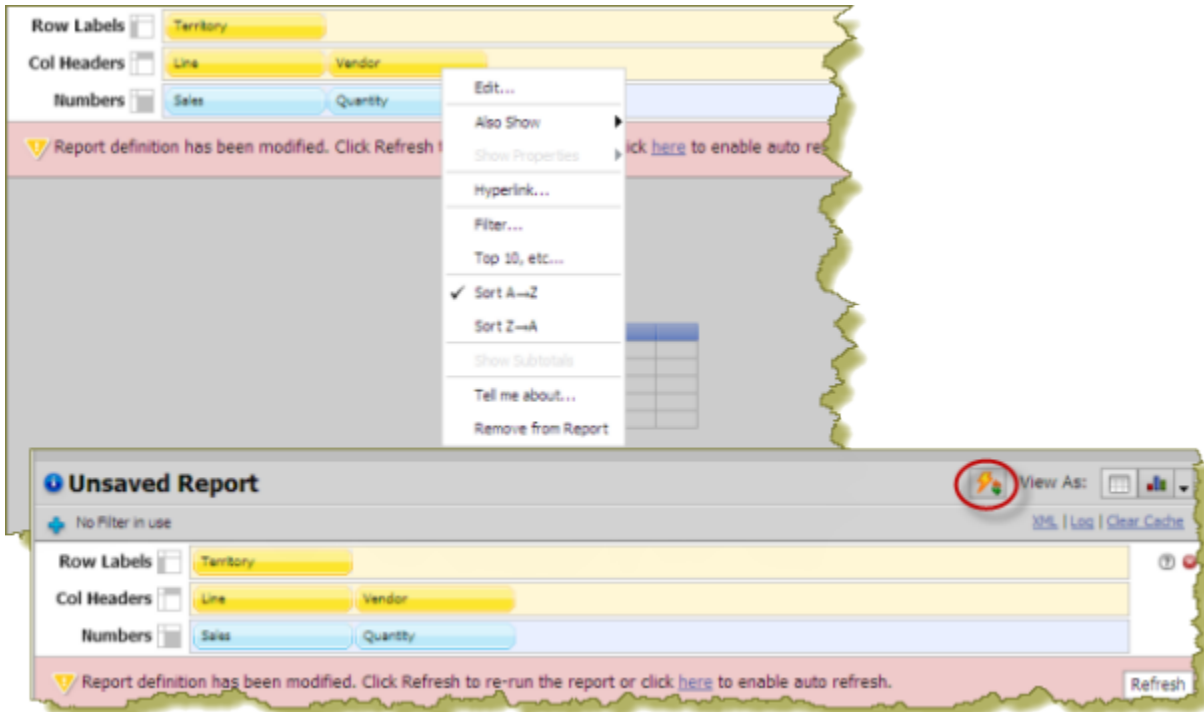




- Click **OK**.
The Analyzer workspace appears.
- To begin creating your report layout, select a field and drag it to the **Field Layout** area at the top of the workspace.

 **Note:** If you are not seeing the Field Layout area, click the **Show Field Layout** button.



A message appears when you place the first field allowing you to refresh the report or to re-enable Auto Refresh. Right-click menus are available for each of the fields you have dragged to the field layout area. You can continue to build your report and only query the database when the layout is complete. You will build your report in the field layout area exclusively while Auto Refresh is disabled. The icons next to **Row Labels**, **Col (Column) Headers**, and **Numbers** indicate the position of the fields once they are placed in the report. Also, a color indicator (yellow or blue) allows you to differentiate between the field types. Notice that the Analyzer workspace is disabled (greyed out) if you are not in live query mode.



 **Note:** Click the enable/disable **Auto Refresh** button () in the toolbar to toggle between the two modes.

- When your report is complete, click **Refresh** in the pink band of the field layout area. Alternatively, use the toggle button.

The Analyzer report updates based on the report definition and displays the results. You can continue to edit the report as needed and click **Refresh** to run the query again.

Row Labels	Territory					
Col Headers	Line	Vendor				
Numbers	Sales	Quantity	Unit Sales			
Line	Vendor					
	Classic Metal Creations			Classic Cars		
	Gearbox Collectibles					
Territory	Sales	Quantity	Unit Sales	Sales	Quantity	Unit Sales
EMEA	356,060	2,855	125	290,141	2,369	122
NA	261,427	2,039	128	209,749	1,663	126
APAC	87,288	694	126	62,652	513	122
Japan	37,919	242	157	22,578	170	133

Creating Hyperlinks to Pentaho Solution Repository Content

This feature allows you to create hyperlinks in an Analyzer report that will link to content in the Pentaho Solution Repository.

The instructions below show you how to create hyperlinks to content inside your Pentaho Solution Repository (reports, action sequences, etc.). You must adjust the instructions to work with your own data.

Important: Hyperlinks take precedence over the drill down feature in charts. That means that if a hyperlink has been defined and you click on a bar in a bar chart, for example, the data associated with the hyperlink, (not the drill down chart), displays. See [Drilling in Charts](#) for details about drill down feature in charts.

1. Create a simple Analyzer report.
2. Select a row label or column header, right-click, and choose **Hyperlink**.

Line	2003	2004	2005
	Sales	Sales	Sales
Classic Ca	407	1,838,275	738,738
Motorcycl	220	590,580	286,325
Planes	755	528,928	200,074
Ships	821	375,672	128,178
Trains	802	124,750	36,917
Trucks am	430	531,976	201,875
Vintage C	949	997,560	388,718

Note: In the example above, **Line** is a row label and **Years** is a column header. Under these, are the individual members that belong to that field/level (for example, Classic Cars, Motorcycles, and so on are members of **Line**).

The **Link on (Name of Field Level)** appears.

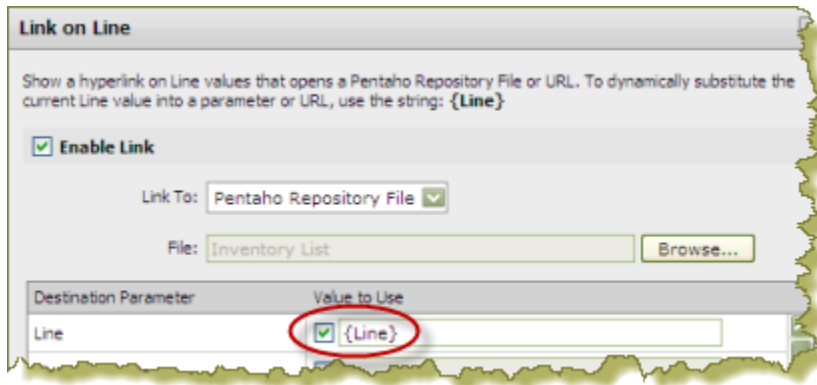
3. Click **Enable Link** to activate the hyperlink feature.

You are not creating a hyperlink to the field header itself but rather to the members under the field level. You can disable linking at any time by clearing the Enable Link checkbox.

4. Click the down arrow next to **Link to** and choose **Pentaho Repository File**.

5. If you choose Pentaho Repository File, you can link to a report (.prpt), an action sequence, or any other file inside the Pentaho Solution Repository which contains a parameter that can be associated with the content in your Analyzer report. Click **Browse** to locate your file in the Pentaho Repository. When you have found the file click **Open**.

The **Link on Field Name** dialog box appears. If a destination parameter that matches the field name in your Analyzer Report is found, field name is associated to that parameter. Click the checkbox next to the destination parameter name to enable it.



If there are more parameters available for linking in the report or other content, they are also listed. You can override the values to the available parameters or leave them as is. You can also define how hyperlink content will display (Open in **New Tab**, **Window**, **Current Window**). Finally, you can enter a **Tool Tip** that displays when users hover over the hyperlinks.

6. Click **OK**.

Hyperlinks appear in your Analyzer report. Click on the links to make sure the content associated with them appears correctly.


Line	Years		
	2003	2004	2005
Classic Cars	1,514,407	1,838,275	738,738
Motorcycles	397,220	590,580	286,325
Planes	347,755	528,928	200,074
Ships	244,821	375,672	128,178
Trains	72,802	124,750	36,917
Trucks and Buses	420,430	531,976	201,875
Vintage Cars	679,949	997,560	388,718

7. Save your Analyzer Report.

Creating Hyperlinks to a URL

This feature allows you to create hyperlinks in an Analyzer report that will link to a URL for system to system linking, for example.

The instructions below show you how to create hyperlinks to a URL. You must adjust the instructions to work with your own data.

 **Important:** Hyperlinks take precedence over the drill down feature in charts. That means that if a hyperlink has been defined and you click on a bar in a bar chart, for example, the data associated with the hyperlink, (not the drill down chart), displays. See [Drilling in Charts](#) for details about drill down feature in charts.

1. Create a simple Analyzer report.
2. Select a row label (or column), right-click, and choose **Hyperlink**. In the example below, **Line** is row label and **Years** is a column header. Under these, are the individual members that belong to that field/level; for example, Classic Cars, Motorcycles, and so on are members of **Line**.

	Years		
Line	2003	2004	2005
	Sales	Sales	Sales
Classic Cars	1,514,407	1,838,275	738,738
Motorcycles	397,220	590,580	286,325
Planes	347,755	528,928	200,074
Ships	244,821	375,672	128,178
Trains	72,802	124,750	36,917
Trucks and Buses	420,430	531,976	201,875
Vintage Cars	679,949	997,560	388,718

The **Link on (Name of Field)** appears.

3. Click **Enable Link** to activate the hyperlink feature.

You are not creating a hyperlink to the field header itself but rather to the members under the field. You can disable linking at any time by clearing the Enable Link checkbox.

4. Click the down arrow next to **Link to** and choose **URL**.

The **Link on Field Name** appears.

5. Enter the URL path. You must use the *{FieldNameInsideCurlyBraces}* format to pass a parameter in your URL as shown in the example below where **{Years}** is the parameter:

```
http://www.google.com/search?query={Years}
```

You can also define how hyperlink content will display (Open in **New Tab, Window, Current Window**). Finally, you can enter a tool tip that displays when users hover over the hyperlinks.

6. Click **OK** to return to your report.

Hyperlinks appear in your Analyzer report. Click on the links to make sure the content associated with them appears correctly.

	Years		
Line	2003	2004	2005
	Sales	Sales	Sales
Classic Cars	1,514,407	1,838,275	738,738
Motorcycles	397,220	590,580	286,325
Planes	347,755	528,928	200,074
Ships	244,821	375,672	128,178
Trains	72,802	124,750	36,917
Trucks and Buses	420,430	531,976	201,875
Vintage Cars	679,949	997,560	388,718

Number Formatting

Analyzer can format number fields in a few different ways:

- **Default:** Uses the cell formatter specified in the Mondrian schema
- **General Number:** A number that can be separated by commas and decimal places, but has no sign. Pound symbols represent number places if they exist, and zeroes represent potential holders for places that do not exist. So for a definition of **00#,###.00**, the number 11223.4 would be represented in Analyzer as 011,223.40.
- **Percentage:** A number that can be separated by commas and decimal places, followed by a percent sign. Pound symbols represent number places if they exist, and zeroes represent potential holders for places that do not exist. So for a definition of **00#,###.00%**, the number 11223.4 would be represented in Analyzer as 011,223.40%.
- **Currency:** A number that can be separated by commas and decimal places, preceded by a currency sign. Pound symbols represent number places if they exist, and zeroes represent potential holders for places that do not exist. So for a definition of **\$00#,###.00**, the number 11223.4 would be represented in Analyzer as \$011,223.40.

- **Expression:** A custom format defined by some user-supplied logic. See [Advanced Conditional Formatting With MDX Expressions](#) on page 46 for more information.

To change the formatting of a measure, right-click on it in the grid, then select **Column Name and Format...** from the context menu.



Note: Formatting options apply to both the report viewer and exports (PDF, XLS).

Conditional Formatting

Conditional formatting in the Analyzer data grid means that cells will be physically affected by the data they contain. The most common form of conditional formatting is stoplight reporting, where cell backgrounds are colored red, green, or yellow depending on user-defined thresholds. Analyzer offers some simple pre-defined methods of conditionally altering visual cues for numeric data in a variety of formats, and the ability to use a custom MDX expression to implement fine-grained conditions.

Simple Conditional Formatting of Measures

Conditional formatting in the Analyzer data grid means that cells will be physically affected by the data they contain. The most common form of conditional formatting is stoplight reporting, where cell backgrounds are colored red, green, or yellow depending on user-defined thresholds. Analyzer offers some simple pre-defined methods of conditionally formatting numeric data. Follow the directions below to implement conditional cell formatting.

1. Right-click a measure in the grid, then select **Conditional Formatting** from the context menu.
A sub-menu with conditional formatting types will appear.
2. Select your preferred number format from the list.
Consult [Conditional Formatting Types](#) on page 46 for more information on simple conditional formatting types.

The analyzer report will refresh and apply the formatting choice you specified.

Conditional Formatting Types

Indicator Type	Description
Color scale	The background cell color will be shaded according to the value of the cell relative to the highest and lowest recorded values in that measure. There are several color progressions to choose from.
Data bar	The cell background is partially filled with a solid color proportional to the scale of the cell's value relative to the highest and lowest recorded values in that measure.
Trend arrow	An upward or downward arrow is displayed to the right of the cell value depending on whether it contains a positive or negative value.

Advanced Conditional Formatting With MDX Expressions

If the premade conditional formatting options in Analyzer are not precise enough for your needs, you can apply custom formatting to a measure by using an MDX expression, as explained below.

1. Right-click a measure in the grid, then select **Column Name and Format...** from the context menu.
The **Edit Column** dialogue will appear.
2. Select **Expression** from the **Format** drop-down list.
A default MDX expression that prints green or red arrows in cells if their values are greater than or less than zero, respectively, will appear in the **MDX Format Expression** field:

```
Case
When [Measures].CurrentMember > 0
Then '#,##0|arrow=up'
When [Measures].CurrentMember < 0
```

```
Then ' |#,##0|arrow=down'
Else ' |#,##0'
End
```

3. Modify the expression to suit your needs. Consult [Conditional Formatting Expressions](#) on page 47 for more information on conditional formatting syntax and options.



Note: If the MDX expression is invalid, an **invalid report definition** error will appear at the top of the dialogue.

4. Click **OK** to commit the change.

The analyzer report will refresh and apply the formatting choices you specified.

Conditional Formatting Expressions

Valid MDX format strings contain properties that apply special rendering for the HTML pivot tables. A format string follows this syntax:

```
| #,### | style = red
```

The leading | (pipe) tells Analyzer that this format string contains properties. The **#,###** is the measure's value format, which in this example is one number separated from three further places or decimal places by a comma (for instance: 3,667). **style=red** is a key/value pair; all possible keys and values are explained in the table below.

The following properties are supported by Analyzer:

Indicator Type	Description	Values
style	Changes the cell background color.	red, yellow, green
arrow	Shows a trend arrow that points up or down. A value of none will not render the arrow; this is useful in situations in which you only want to show one directional arrow instead of two.	up, down, none
link	Creates an HTML link which will open in a new window when clicked in Analyzer.	Any browser-renderable URL
image	Renders a custom image that you specify. This image file must be stored in the <code>/pentaho-solutions/system/analyzer/resource/image/report/</code> directory.	An image file name, with extension

Using Analyzer on an Apple iPad

Pentaho supports the Analyzer client tool on the Apple iPad (both first- and second-generation). However, the rest of the Pentaho User Console and other Web-based client tools such as Interactive Reporting and Dashboard Designer are not yet supported. In order to safely bypass the Pentaho User Console and go directly to Analyzer, you must use this URL on your iPad instead of the standard User Console address (replacing **example.com** and **8080** with the hostname and port of your BI Server instance):

```
http://example.com:8080/pentaho/content/analyzer/selectSchema
```

When you go to this URL on your iPad, you'll be redirected to the Pentaho User Console login screen. Type in your user credentials (be careful of the auto-correct feature, which may incorrectly capitalize your username) and click **Log in**, and your browser will go directly to Analyzer. You should bookmark and/or add this URL to your Home screen.

Analyzer does not have any reduced functionality on the iPad -- you can create new content and open existing content without any trouble. However, new Analyzer reports are limited to the existing data sources that you've defined through the Data Source Wizard. The wizard, like the rest of the Pentaho User Console, is not supported on the iPad.



Note: You will not be able to schedule Analyzer content from your iPad. If you need to set up a schedule, just save your Analyzer report to the BI Server solution repository and log in from a normal desktop or laptop computer at a later time.

Here are some specific notes on Analyzer interactivity and usability on an iPad:

- You must wait for Analyzer to completely load before interacting with it. You'll know Analyzer is loaded when the browser bar at the top does not show a blue loading indicator

- Use a long press (> 200ms) to open right-click context menus. Available on field list items, pivot table members, pivot table headers, gems, and filter summary items
- Long presses are less sensitive than normal so that you don't have to have extremely steady fingers to open the context menu
- Use a double-tap to add fields to the report (similar to normal double-click behavior on a desktop computer)
- Use the two-finger scroll gesture to scroll field lists and pivot tables
- Drag-and-Drop is less sensitive than normal so that it doesn't get confused with a long press
- You can drag and drop a field from the grid to the Available Fields list to remove it from the report
- Analyzer field list tooltips are disabled, since this would interfere with other options
- One-touch scrolling and scaling is disabled

Creating a New Analysis View



Note: JPivot is deprecated; Pentaho Analyzer is the more modern Enterprise Edition replacement.

To create a new analysis view, follow this process.

1. In the **File** menu, select **New Analysis View...**

This is one of several ways to create a new analysis view; all methods lead to the same screen in the Pentaho User Console.

2. Select the appropriate schema and cube from the drop-down lists.

3. Click **OK** to continue.

A fresh analysis view will open in a new tab.

4. Click the **+** icons next to each column you want to drill down into.

5. To change the filter, column, and row data, click the **Open OLAP Navigator** button, on the left end of the toolbar.

The OLAP Navigator will appear above your analysis view.

6. To add a dynamic chart, click the **Chart** button toward the right side of the toolbar.

A new chart will appear below your analysis view. It will change dynamically according to the drill-down you select.

7. When you're finished exploring your data through an analysis view, you can selecting **Save** from the **File** menu, or by clicking the disk icon in the left side of the toolbar.

You now have an analysis view, which enables you to drill down to the minutest of details stored in your database. Using this analysis view, you can create an interactive dashboard, or share your findings with other Pentaho User Console users.

Using Analysis Cubes in Report Designer

Pentaho Report Designer is typically used with raw data sources, or with databases that have a metadata layer through a tool like Pentaho Metadata Editor. However, you can also use an MDX query to retrieve a data set from a multidimensional database. The sections below explain how to set up a Pentaho Analysis data source and add an MDX query in Pentaho Report Designer.



Note: For more information on Report Designer, consult the Report Designer User Guide in the Pentaho Knowledge Base, or through the **Documentation** link in the **Help** menu in Report Designer.

Adding an OLAP Data Source

You must have a report file open in order to proceed, and your data source must be accessible before you can connect to it in Report Designer. You may need to obtain database connection information from your system administrator, such as the URL, port number, JDBC connection string, database type, and user credentials.

Follow this procedure to add a Pentaho Analysis (Mondrian) data source in Report Designer.

1. Select the **Data** tab in the upper right pane.

By default, Report Designer starts in the **Structure** tab, which shares a pane with **Data**.

2. Click the yellow cylinder icon in the upper left part of the Data pane, or right-click **Data Sets**.

A drop-down menu with a list of supported data source types will appear.

3. Select **OLAP** from the drop-down menu, then select one of the following: **Pentaho Analysis**, **Pentaho Analysis (Denormalized)**, or **Pentaho Analysis (Legacy)**.
The **Mondrian Datasource Editor** window will appear.
4. If you want to provide parameters that contain different Mondrian connection authentication credentials, click the **Edit Security** button in the upper left corner of the window, then type in the fields or variables that contain the user credentials you want to store as a parameter with this connection.
The role, username, and password will be available as a security parameter when you are creating your report.
5. Click **Browse**, navigate to your Mondrian schema XML file, then click **Open**.
6. Above the **Connections** pane on the left, click the round green **+** icon to add a new data source.
If you installed the Pentaho sample data, several **SampleData** entries will appear in the list. These sample data sources are useless if you do not have the Pentaho HSQLDB sample database installed, so if you don't have that, you can safely delete the SampleData entries. If you do have Pentaho's HSQLDB samples installed, it may be advantageous to leave the sample data sources intact in the event that you want to view the sample reports and charts at a later time.
7. In the subsequent **Database Connection** dialogue, type in a concise but reasonably descriptive name for this connection in the **Connection Name** field; select your database brand from the **Connection Type** list; select the access type in the **Access** list at the bottom; then type in your database connection details into the fields in the **Settings** section on the right.
The Access list will change according to the connection type you select; the settings section will change depending on which item in the access list you choose.
8. Click the **Test** button to ensure that the connection settings are correct. If they are not, the ensuing error message should give you some clues as to which settings need to be changed. If the test dialogue says that the connection to the database is OK, then click the **OK** button to complete the data source configuration.

Now that your data source is configured, you must enter an MDX query before you can finish adding the data source. See [Creating an MDX Query](#) on page 49 for details.

Creating an MDX Query

You must be in the Data Source Configuration window to follow this process. You should also have configured a JNDI data source connection.

Follow this process to add a Mondrian cube definition (multidimensional expression) as a data source.

1. At the bottom of the Data Source Configuration window, click **Use Mondrian Cube Definition (MDX)**.
The **Browse** button will become active.
2. Click **Browse**, navigate to the location of your cube definition file, click to select it, then click **Open**.
3. Type an MDX query into the **Query** pane on the right.
4. Click **OK**.

Your data source is now properly configured and refined according to the Mondrian definition you specified.

Logging

The BI Server and the standalone Mondrian engine both use **log4j** to record information about generated ROLAP-generated SQL queries. It may be useful to examine this log output if you are having trouble with configuration or performance-tuning.

Analysis SQL Output Logging

Follow the directions below to turn on logging for SQL generation in the BI Server or Mondrian engine. If you are aware of other log4j classes that will assist you in configuration troubleshooting or performance-tuning, you can add them during this process.

1. Stop the BI Server or standalone Mondrian engine.
2. Edit the **log4j.xml** file in the `/WEB-INF/classes/` directory inside of the deployed pentaho.war or Mondrian engine.



Note: In some Mondrian implementations, this file is **log4j.properties**. Substitute this filename accordingly if you do not see a log4j.xml file.

3. Add the following line anywhere inside of the `</log4j:configuration>` node (but not inside of other constituent node tags):

```
log4j.logger.mondrian.sql=DEBUG
```

4. Save and close the file.
5. Start the BI Server or Mondrian engine.

A significant amount of Analysis SQL query debugging information will now be available in your application server's logs. Some data about the segment cache plugin will also appear, but there are some other steps to take beyond this if you want log data specific to the segment cache. Refer to [Enabling Segment Cache Logging](#) on page 50 for more information.

Enabling Segment Cache Logging

When deployed inside or alongside of the Pentaho BI Server, you can view Analysis engine segment cache logs by opening the log window from within Analyzer.

If you are using Analyzer with a standalone Mondrian engine, cache log information is available through log4j. Edit your log4j configuration as explained in [Analysis SQL Output Logging](#) on page 50, and set the value of **com.pentaho.analysis.segmentcache** to DEBUG. Here are some other log4j categories that can help you diagnose configuration and performance problems:

Class name	Cache system
com.pentaho.analysis.segmentcache.impl.infinispan	Outputs information related to the Infinispan implementation of the segment cache.
com.pentaho.analysis.segmentcache.impl.memcached	Outputs information related to the Memcached implementation of the segment cache.
com.pentaho.analysis.segmentcache.impl.pentaho	Outputs information related to the Pentaho BI Platform Delegating Cache implementation.

Viewing Log Output in Analyzer

Create a new Pentaho Analyzer report and drag a measure or a dimension in the report. Once your report is displaying numbers, click on the top link, identified as "Log". If there is no such link available within Pentaho Analyzer, it is because you have not used an administrator account to login to Pentaho User Console. You must log out and log back in using an administrator account credentials.

If your plugin was successfully configured, you should see log information relating to the segment cache. If there was an error in your configuration, all exception messages will be displayed here as well.

Troubleshooting

This section contains known problems and solutions relating to the procedures covered in this guide.

Old Analysis Schemas Still Show Up in Pentaho User Console

If you have unwanted analysis schemas in the schema list in the Pentaho User Console -- this is the list that appears when you create a new analysis view or Analyzer report -- then you must edit the `datasources.xml` file and remove the unwanted entries as explained in [Removing Mondrian Data Sources](#) on page 15.

Multi-Byte Characters Don't Appear In PDFs Exported From Analyzer

If you are using a multi-byte character set, such as would be used for languages like Japanese or Chinese, and find that you have missing or corrupted output when exporting Analyzer reports to PDF, you will have to specify a default TrueType font for PDF rendering that supports multi-byte characters. The default PDF font in Analyzer is Helvetica, which does not support multi-byte character sets. To change this setting, see [Setting a Default Font for PDF Exports](#) on page 22.

Mondrian Schema Element Quick Reference

All of the possible Mondrian schema elements are defined in brief below, listed in the hierarchy in which they are used. To see a more detailed definition and a list of possible attributes and content, click on an element name.

Element	Definition
<Schema>	A complete Mondrian schema; a collection of cubes, virtual cubes, shared dimensions, and roles.
<Cube>	A collection of dimensions and measures, all centered on a fact table.
<VirtualCube>	A cube defined by combining the dimensions and measures of one or more cubes. A measure originating from another cube can be a <CalculatedMember> .
<CubeUsages>	Base cubes that are imported into a virtual cube.
<CubeUsage>	Usage of a base cube by a virtual cube.
<VirtualCubeDimension>	Usage of a dimension by a virtual cube.
<VirtualCubeMeasure>	Usage of a measure by a virtual cube.
<Dimension>	Defines a dimension -- a collection of hierarchies.
<DimensionUsage>	Usage of a shared dimension by a cube.
<Hierarchy>	Specifies a predefined drill-down.
<Level>	A level of a hierarchy.
<KeyExpression>	SQL expression used as key of the level, in lieu of a column.
<NameExpression>	SQL expression used to compute the name of a member, in lieu of Level.nameColumn.
<CaptionExpression>	SQL expression used to compute the caption of a member, in lieu of Level.captionColumn.
<OrdinalExpression>	SQL expression used to sort members of a level, in lieu of Level.ordinalColumn.
<ParentExpression>	SQL expression used to compute a measure, in lieu of Level.parentColumn.
<Property>	A member property. The definition is contained in a hierarchy or level, but the property will be available to all members.
<PropertyExpression>	SQL expression used to compute the value of a property, in lieu of Property.column.
<Measure>	Specifies an aggregated numeric value.
<CalculatedMember>	A member whose value is derived using a formula, defined as part of a cube.
<NamedSet>	A set whose value is derived using a formula, defined as part of a cube.
<Table>	A fact or dimension table.
<View>	Defines a table by using an SQL query, which can have different variants for different underlying databases.

Element	Definition
<Join>	Defines a table by joining a set of queries.
<InlineTable>	Defines a table using an inline dataset.
<Closure>	Maps a parent-child hierarchy onto a closure table.
<AggExclude>	Exclude a candidate aggregate table by name or pattern matching.
<AggName>	Declares an aggregate table to be matched by name.
<AggPattern>	Declares a set of aggregate tables by regular expression pattern.
<AggFactCount>	Specifies name of the column in the candidate aggregate table which contains the number of fact table rows.
<AggIgnoreColumn>	Tells Mondrian to ignore a column in an aggregate table.
<AggForeignKey>	Maps a foreign key in the fact table to a foreign key column in the candidate aggregate table.
<AggMeasure>	Maps a measure to a column in the candidate aggregate table.
<AggLevel>	Maps a level to a column in the candidate aggregate table.
<Role>	An access-control profile.
<SchemaGrant>	A set of rights to a schema.
<CubeGrant>	A set of rights to a cube.
<HierarchyGrant>	A set of rights to both a hierarchy and levels within that hierarchy.
<MemberGrant>	A set of rights to a member and its children.
<Union>	Definition of a set of rights as the union of a set of roles.
<RoleUsage>	A reference to a role.
<UserDefinedFunction>	Imports a user-defined function.
<Parameter>	Part of the definition of a hierarchy; passed to a MemberReader, if present.
<CalculatedMemberProperty>	Property of a calculated member.
<Formula>	Holds the formula text within a <NamedSet> or <CalculatedMember>.
<ColumnDefs>	Holder for <ColumnDef> elements.
<ColumnDef>	Definition of a column in an <InlineTable> dataset.
<Rows>	Holder for <Row> elements.
<Row>	Row in an <InlineTable> dataset.
<Value>	Value of a column in an <InlineTable> dataset.
<MeasureExpression>	SQL expression used to compute a measure, in lieu of Measure.column.
<SQL>	The SQL expression for a particular database dialect.

AggExclude

This element excludes a candidate aggregate table via name or pattern matching.

Attributes

Attribute	Data type	Definition
pattern	String	A Table pattern not to be matched.
name	String	The Table name not to be matched.
ignorecase	Boolean	Whether or not the match should ignore case.

AggFactCount

Specifies name of the column in the candidate aggregate table which contains the number of fact table rows.

Attributes

Attribute	Data type	Definition
N/A	N/A	N/A

AggForeignKey

Maps foreign key in the fact table to a foreign key column in the candidate aggregate table.

Attributes

Attribute	Data type	Definition
factColumn	String	The name of the base fact table foreign key.
aggColumn	String	The name of the aggregate table foreign key.

AggIgnoreColumn

Tells Mondrian to ignore a column in an aggregate table.

Attributes

Attribute	Data type	Definition
N/A	N/A	N/A

AggLevel

Maps a level to a column in the candidate aggregate table.

Attributes

Attribute	Data type	Definition
column	String	The name of the column mapping to the level name.
name	String	The name of the Dimension Hierarchy level.

AggMeasure

Maps a measure to a column in the candidate aggregate table.

Attributes

Attribute	Data type	Definition
column	String	The name of the column mapping to the measure name.
name	String	The name of the Cube measure.

AggName

Declares an aggregate table to be matched by name.

Attributes

Attribute	Data type	Definition
name	String	The table name of a specific aggregate table.

AggPattern

Declares a set of aggregate tables by regular expression pattern.

Attributes

Attribute	Data type	Definition
pattern	String	A Table pattern used to define a set of aggregate tables.

Constituent elements

Element	Definition
AggExclude	

AggTable

A definition of an aggregate table for a base fact table. This aggregate table must be in the same schema as the base fact table.

Attributes

Attribute	Data type	Definition
ignorecase	Boolean	Whether or not the match should ignore case.

Constituent elements

Element	Definition
AggFactCount	Describes what the fact_count column looks like.
AggIgnoreColumn	
AggForeignKey	
AggMeasure	
AggLevel	

CalculatedMember

A member whose value is derived using a formula, defined as part of a cube.

Attributes

Attribute	Data type	Definition
name	String	Name of this calculated member.
formatString	String	Format string with which to format cells of this member. For more details, see {@link mondrian.util.Format} .
caption	String	A string being displayed instead of the name. Can be localized from Properties file using <code>#{propertyname}</code> .
formula	String	MDX expression which gives the value of this member. Equivalent to the Formula sub-element.
dimension	String	Name of the dimension which this member belongs to.
visible	Boolean	Whether this member is visible in the user-interface. Default true.

Constituent elements

Element	Definition
Formula	MDX expression which gives the value of this member.
CalculatedMemberProperty	

CalculatedMemberProperty

Property of a calculated member defined against a cube. It must have either an expression or a value.

Attributes

Attribute	Data type	Definition
name	String	Name of this member property.
caption	String	A string being displayed instead of the Properties's name. Can be localized from Properties file using <code>#{propertyname}</code> .
expression	String	MDX expression which defines the value of this property. If the expression is a constant string, you could enclose it in quotes, or just specify the 'value' attribute instead.
value	String	Value of this property. If the value is not constant, specify the 'expression' attribute instead.

CaptionExpression

SQL expression used to compute the caption of a member, in lieu of `Level.captionColumn`.

Attributes

Attribute	Data type	Definition
N/A	N/A	N/A

Closure

Specifies the transitive closure of a parent-child hierarchy. Optional, but recommended for better performance. The closure is provided as a set of (parent/child) pairs: since it is the transitive closure these are actually (ancestor/descendant) pairs.

Attributes

Attribute	Data type	Definition
parentColumn	String	The parent column
childColumn	String	The child column

Constituent elements

Element	Definition
Table	

ColumnDef

Column definition for an inline table.

Attributes

Attribute	Data type	Definition
name	String	Name of the column.
type	String	Type of the column: String, Numeric, Integer, Boolean, Date, Time or Timestamp.

ColumnDefs

Holder for an array of ColumnDef elements.

Constituent elements

Element	Definition
ColumnDef	The columns to include in the array

Cube

A cube is a collection of dimensions and measures, all centered on a fact table.

Attributes

Attribute	Data type	Definition
name	String	Name of this cube.
caption	String	A string being displayed instead of the cube's name. Can be localized from Properties file using #{propertyname}.
defaultMeasure	String	The name of the measure that would be taken as the default measure of the cube.
cache	Boolean	Should the Fact table data for this Cube be cached by Mondrian or not. The default action is to cache the data.

Attribute	Data type	Definition
enabled	Boolean	Whether element is enabled - if true, then the Cube is realized otherwise it is ignored.

Constituent elements

Element	Definition
Relation (as <Table> , <View> , or <InlineTable>)	The fact table is the source of all measures in this cube. If this is a Table and the schema name is not present, table name is left unqualified.
CubeDimension	
Measure	
CalculatedMember	Calculated members in this cube.
NamedSet	Named sets in this cube.

CubeGrant

Grants (or denies) this role access to a cube. access may be "all" or "none".

Attributes

Attribute	Data type	Definition
cube	String	The unique name of the cube

Constituent elements

Element	Definition
DimensionGrant	
HierarchyGrant	

CubeUsage

Usage of a base cube by a virtual cube.

Attributes

Attribute	Data type	Definition
cubeName	String	Name of the cube which the virtualCube uses.
ignoreUnrelatedDimensions	Boolean	Unrelated dimensions to measures in this cube will be pushed to top level member.

CubeUsages

List of base cubes used by the virtual cube.

Constituent elements

Element	Definition
CubeUsage	

Dimension

A Dimension is a collection of hierarchies. There are two kinds: a public dimension belongs to a Schema, and be used by several cubes; a private dimension belongs to a Cube. The foreignKey field is only applicable to private dimensions.

Attributes

Attribute	Data type	Definition
name	String	The name of this dimension.
type	String	The dimension's type may be one of "Standard" or "Time". A time dimension will allow the use of the MDX time functions (WTD, YTD, QTD, etc.). Use a standard dimension if the dimension is not a time-related dimension. The default value is "Standard".
caption	String	A string being displayed instead of the dimensions's name. Can be localized from Properties file using #{propertyname}.
usagePrefix	String	If present, then this is prepended to the Dimension column names during the building of collapse dimension aggregates allowing 1) different dimensions to be disambiguated during aggregate table recognition. This should only be set for private dimensions.

Constituent elements

Element	Definition
Hierarchy	The hierarchies (pre-defined drill-downs) for this dimension.

DimensionGrant

Grants (or denies) this role access to a dimension. access may be "all" or "none". Note that a role is implicitly given access to a dimension when it is given access to a cube. See also the "all_dimensions" option of the "SchemaGrant" element.

Attributes

Attribute	Data type	Definition
dimension	String	The unique name of the dimension

DimensionUsage

A DimensionUsage is usage of a shared Dimension within the context of a cube.

Attributes

Attribute	Data type	Definition
source	String	Name of the source dimension. Must be a dimension in this schema. Case-sensitive.

Attribute	Data type	Definition
level	String	Name of the level to join to. If not specified, joins to the lowest level of the dimension.
usagePrefix	String	If present, then this is prepended to the Dimension column names during the building of collapse dimension aggregates allowing 1) different dimension usages to be disambiguated during aggregate table recognition and 2) multiple shared dimensions that have common column names to be disambiguated.

Formula


Holds the formula text within a <NamedSet> or <CalculatedMember>.

Attributes

Attribute	Data type	Definition
N/A	N/A	N/A

Hierarchy

Defines a hierarchy, which is a pre-defined drill-down.

 **Note:** You must specify at most one <Relation> or memberReaderClass. If you specify none, the hierarchy is assumed to come from the same fact table of the current cube.

Attributes

Attribute	Data type	Definition
name	String	Name of the hierarchy. If this is not specified, the hierarchy has the same name as its dimension.
hasAll	Boolean	Whether this hierarchy has an 'all' member.
allMemberName	String	Name of the 'all' member. If this attribute is not specified, the all member is named 'All hierarchyName', for example, 'All Store'.
allMemberCaption	String	A string being displayed instead as the all member's name. Can be localized from Properties file using #{propertyname}.
allLevelName	String	Name of the 'all' level. If this attribute is not specified, the all member is named '(All)'. Can be localized from Properties file using #{propertyname}.
primaryKey	String	The name of the column which identifies members, and which is referenced by rows in the fact table. If not specified, the key of the lowest level is used. See also CubeDimension.foreignKey.

Attribute	Data type	Definition
primaryKeyTable	String	The name of the table which contains primaryKey. If the hierarchy has only one table, defaults to that; it is required.
defaultMember	String	
memberReaderClass	String	Name of the custom member reader class. Must implement the mondrian.rolap.MemberReader interface.
caption	String	A string to be displayed in the user interface. If not specified, the hierarchy's name is used. Can be localized from Properties file using #{propertyname}.

Constituent elements

Element	Definition
RelationOrJoin (as <Table> , <View> , <Join> , or <InlineTable>)	The Table, Join, View, or Inline Table that populates this hierarchy.
Level	

HierarchyGrant

Grants (or denies) this role access to a hierarchy. access may be "all", "custom" or "none". If access is "custom", you may also specify the attributes topLevel, bottomLevel, and the member grants.

Attributes

Attribute	Data type	Definition
hierarchy	String	The unique name of the hierarchy
topLevel	String	Unique name of the highest level of the hierarchy from which this role is allowed to see members. May only be specified if the HierarchyGrant.access is "custom". If not specified, role can see members up to the top level.
bottomLevel	String	Unique name of the lowest level of the hierarchy from which this role is allowed to see members. May only be specified if the HierarchyGrant.access is "custom". If not specified, role can see members down to the leaf level.
rollupPolicy	String	Policy which determines how cell values are calculated if not all of the children of the current cell are visible to the current role. Allowable values are 'full' (the default), 'partial', and 'hidden'.

Constituent elements

Element	Definition
MemberGrant	

InlineTable

Defines a table using an inline dataset.

Attributes

Attribute	Data type	Definition
alias	String	Alias to be used with this table when it is used to form queries. If not specified, defaults to the table name, but in any case, must be unique within the schema. (You can use the same table in different hierarchies, but it must have different aliases.)

Constituent elements

Element	Definition
ColumnDefs	
Rows	

Join

Defines a 'table' by joining a set of queries.

Attributes

Attribute	Data type	Definition
leftAlias	String	Defaults to left's alias if left is a table, otherwise required.
leftKey	String	
rightAlias	String	Defaults to right's alias if right is a table, otherwise required.
rightKey	String	

Constituent elements

Element	Definition
RelationOrJoin (as <Table> , <View> , <Join> , or <InlineTable>)	There must be two defined; a left , and a right

KeyExpression

SQL expression used as key of the level, in lieu of a column.

Attributes

Attribute	Data type	Definition
N/A	N/A	N/A

Level

Level of a hierarchy.

Attributes

Attribute	Data type	Definition
approxRowCount	String	The estimated number of members in this level. Setting this property improves the performance of MDSHEMA_LEVELS,

Attribute	Data type	Definition
		MDSHEMA_HIERARCHIES and MDSHEMA_DIMENSIONS XMLA requests
name	String	
table	String	The name of the table that the column comes from. If this hierarchy is based upon just one table, defaults to the name of that table; otherwise, it is required. Can be localized from Properties file using #{propertyname}.
column	String	The name of the column which holds the unique identifier of this level.
nameColumn	String	The name of the column which holds the user identifier of this level.
ordinalColumn	String	The name of the column which holds member ordinals. If this column is not specified, the key column is used for ordering.
parentColumn	String	The name of the column which references the parent member in a parent-child hierarchy.
nullParentValue	String	Value which identifies null parents in a parent-child hierarchy. Typical values are 'NULL' and '0'.
type	String	Indicates the type of this level's key column: String, Numeric, Integer, Boolean, Date, Time or Timestamp. When generating SQL statements, Mondrian encloses values for String columns in quotation marks, but leaves values for Integer and Numeric columns un-quoted. Date, Time, and Timestamp values are quoted according to the SQL dialect. For a SQL-compliant dialect, the values appear prefixed by their typename, for example, "DATE '2006-06-01'".
uniqueMembers	Boolean	Whether members are unique across all parents. For example, zipcodes are unique across all states. The first level's members are always unique.
levelType	String	Whether this is a regular or a time-related level. The value makes a difference to time-related functions such as YTD (year-to-date).
hideMemberIf	String	Condition which determines whether a member of this level is hidden. If a hierarchy has one or more levels with hidden members, then it is possible that not all leaf members are the same distance from the root, and it is termed a ragged hierarchy.
formatter	String	Name of a formatter class for the member labels being displayed. The class must implement the mondrian.olap.MemberFormatter interface. Allowable values are: Never (a member always appears; the default); IfBlankName (a member

Attribute	Data type	Definition
		doesn't appear if its name is null or empty); and IfParentsName (a member appears unless its name matches the parent's).
caption	String	A string being displayed instead of the level's name. Can be localized from Properties file using #{propertyname}.
captionColumn	String	The name of the column which holds the caption for members.

Constituent elements

Element	Definition
KeyExpression	The SQL expression used to populate this level's key.
NameExpression	The SQL expression used to populate this level's name. If not specified, the level's key is used.
OrdinalExpression	The SQL expression used to populate this level's ordinal.
ParentExpression	The SQL expression used to join to the parent member in a parent-child hierarchy.
Closure	
Property	

Measure

A measure is an aggregated numeric value. Typically it is a sum of selected numbers in a column, or a count of the number of items in a list.

Attributes

Attribute	Data type	Definition
name	String	Name of this measure.
column	String	Column which is source of this measure's values. If not specified, a measure expression must be specified.
datatype	String	The datatype of this measure: String, Numeric, Integer, Boolean, Date, Time or Timestamp. The default datatype of a measure is 'Integer' if the measure's aggregator is 'Count', otherwise it is 'Numeric'.
formatString	String	Format string with which to format cells of this measure. For more details, see the <code>mondrian.util.Format</code> class.
aggregator	String	Aggregation function. Allowed values are "sum", "count", "min", "max", "avg", and "distinct-count". ("distinct count" is allowed for backwards compatibility, but is deprecated because XML enumerated attributes in a DTD cannot legally contain spaces.)
formatter	String	Name of a formatter class for the appropriate cell being displayed.

Attribute	Data type	Definition
		The class must implement the mondrian.olap.CellFormatter interface.
caption	String	A string being displayed instead of the name. Can be localized from Properties file using #{propertyname}.
visible	Boolean	Whether this member is visible in the user-interface. Default true.

Constituent elements

Element	Definition
MeasureExpression	The SQL expression used to calculate a measure. Must be specified if a source column is not specified.
CalculatedMemberProperty	

MeasureExpression

A schema is a collection of cubes and virtual cubes. It can also contain shared dimensions (for use by those cubes), named sets, roles, and declarations of user-defined functions.

Attributes

Attribute	Data type	Definition
-----------	-----------	------------

Constituent elements

Element	Definition
---------	------------

MemberGrant

A schema is a collection of cubes and virtual cubes. It can also contain shared dimensions (for use by those cubes), named sets, roles, and declarations of user-defined functions.

Attributes

Attribute	Data type	Definition
-----------	-----------	------------

Constituent elements

Element	Definition
---------	------------

NamedSet

A schema is a collection of cubes and virtual cubes. It can also contain shared dimensions (for use by those cubes), named sets, roles, and declarations of user-defined functions.

Attributes

Attribute	Data type	Definition
-----------	-----------	------------

Constituent elements

Element	Definition
---------	------------

NameExpression

A schema is a collection of cubes and virtual cubes. It can also contain shared dimensions (for use by those cubes), named sets, roles, and declarations of user-defined functions.

Attributes

Attribute	Data type	Definition
-----------	-----------	------------

Constituent elements

Element	Definition
---------	------------

OrdinalExpression

A schema is a collection of cubes and virtual cubes. It can also contain shared dimensions (for use by those cubes), named sets, roles, and declarations of user-defined functions.

Attributes

Attribute	Data type	Definition
-----------	-----------	------------

Constituent elements

Element	Definition
---------	------------

Parameter

A schema is a collection of cubes and virtual cubes. It can also contain shared dimensions (for use by those cubes), named sets, roles, and declarations of user-defined functions.

Attributes

Attribute	Data type	Definition
-----------	-----------	------------

Constituent elements

Element	Definition
---------	------------

ParentExpression

A schema is a collection of cubes and virtual cubes. It can also contain shared dimensions (for use by those cubes), named sets, roles, and declarations of user-defined functions.

Attributes

Attribute	Data type	Definition
-----------	-----------	------------

Constituent elements

Element	Definition
---------	------------

Property

Member property that enables you to create subcategories for levels and hierarchies.

Attributes

Attribute	Data type	Definition
name	String	The display name of this property.
column	String	The data column that will determine this subcategory's content.
type	String	Data type of this property: String, Numeric, Integer, Boolean, Date, Time or Timestamp.
formatter	String	Name of a formatter class for the appropriate property value being displayed. The class must implement the mondrian.olap.PropertyFormatter interface.
caption	String	A string being displayed instead of the name. Can be localized from Properties file using #{propertyname}.

PropertyExpression

SQL expression used to compute the value of a property, in lieu of Property.column.

Attributes

Attribute	Data type	Definition
N/A	N/A	N/A

Role

A role defines an access-control profile. It has a series of grants (or denials) for schema elements.

Attributes

Attribute	Data type	Definition
name	String	The name of this role

Constituent elements

Element	Definition
SchemaGrant	
Union	

RoleUsage

Usage of a Role in a union Role.

Attributes

Attribute	Data type	Definition
roleName	String	The name of the role

Row

Row definition for an inline table. Must have one Column for each ColumnDef in the InlineTable.

Constituent elements

Element	Definition
Value	

Rows

Holder for an array of Row elements

Constituent elements

Element	Definition
Row	

Schema

A schema is a collection of cubes and virtual cubes. It can also contain shared dimensions (for use by those cubes), named sets, roles, and declarations of user-defined functions.

Attributes

Attribute	Data type	Definition
name	String	Name of this schema
measuresCaption	String	Label for the measures dimension. Can be localized from Properties file using #{propertyname}.
defaultRole	String	The name of the default role for connections to this schema

Constituent elements

Element	Definition
Parameter	This schema's parameter definitions.
Dimension	Shared dimensions in this schema.
Cube	Cubes in this schema.
VirtualCube	Virtual cubes in this schema.
NamedSet	Named sets in this schema.
Role	Roles in this schema.
UserDefinedFunction	Declarations of user-defined functions in this schema.

SchemaGrant

Grants (or denies) this role access to this schema. access may be "all", "all_dimensions", or "none". If access is "all_dimensions", the role has access to all dimensions but still needs explicit access to cubes.

Constituent elements

Element	Definition
CubeGrant	

SQL

The SQL expression for a particular database dialect.

Attributes

Attribute	Data type	Definition
dialect	String	Dialect of SQL the view is intended for. Valid values include, but are not limited to: <ul style="list-style-type: none">• generic• access• db2• derby• firebird• hsqldb• mssql• mysql• oracle• postgres• sybase• teradata• ingres• infobright• luciddb

Table

A fact or dimension table.

Attributes

Attribute	Data type	Definition
name	String	The name of the table
schema	String	Optional qualifier for table.
alias	String	Alias to be used with this table when it is used to form queries. If not specified, defaults to the table name, but in any case, must be unique within the schema. (You can use the same table in different hierarchies, but it must have different aliases.)

Constituent elements

Element	Definition
SQL	The SQL WHERE clause expression to be appended to any select statement
AggExclude	
AggTable	

Union

Body of a Role definition which defines a Role to be the union of several Roles. The RoleUsage elements must refer to Roles that have been declared earlier in this schema file.

Constituent elements

Element	Definition
RoleUsage	

UserDefinedFunction

A UserDefinedFunction is a function which extends the MDX language. It must be implemented by a Java class which implements the interface mondrian.spi.UserDefinedFunction.

Attributes

Attribute	Data type	Definition
name	String	Name with which the user-defined function will be referenced in MDX expressions.
className	String	Name of the class which implements this user-defined function. Must implement the mondrian.spi.UserDefinedFunction interface.

Value

Column value for an inline table. The CDATA holds the value of the column.

Attributes

Attribute	Data type	Definition
column	String	Name of the column.

View

Defines a 'table' using a SQL query, which can have different variants for different underlying databases.

Attributes

Attribute	Data type	Definition
alias	String	

Constituent elements

Element	Definition
SQL	

VirtualCube

A cube defined by combining the dimensions and measures of one or more cubes. A measure originating from another cube can be a <CalculatedMember>.

Attributes

Attribute	Data type	Definition
enabled	Boolean	Whether this element is enabled - if true, then the Virtual Cube is realized otherwise it is ignored.
name	String	
defaultMeasure	String	The name of the measure that would be taken as the default measure of the cube.
caption	String	A string being displayed instead of the cube's name. Can be localized from Properties file using #{propertyname}.

Constituent elements

Element	Definition
CubeUsages	
VirtualCubeDimension	
VirtualCubeMeasure	
CalculatedMember	Calculated members that belong to this virtual cube. (Calculated members inherited from other cubes should not be in this list.)
NamedSet	Named sets in this cube.

VirtualCubeDimension

A VirtualCubeDimension is a usage of a Dimension in a VirtualCube.

Attributes

Attribute	Data type	Definition
cubeName	String	Name of the cube which the dimension belongs to, or unspecified if the dimension is shared.
name	String	Name of the dimension.

VirtualCubeMeasure

A VirtualCubeMeasure is a usage of a Measure in a VirtualCube.

Attributes

Attribute	Data type	Definition
cubeName	String	Name of the cube which the measure belongs to.
name	String	Unique name of the measure within its cube.
visible	Boolean	Whether this member is visible in the user-interface. Default true.