Oracle Business Intelligence 11*g* R1 Cookbook

Make complex analytical reports simple and deliver valuable business data using OBIEE 11*g* with this comprehensive and practical guide

Cuneyt Yilmaz

[PACKT] enterprise⌗
professional expertise distilled
PUBLISHING

Oracle Business Intelligence 11*g* R1 Cookbook

Make complex analytical reports simple and deliver valuable business data using OBIEE 11*g* with this comprehensive and practical guide

Cuneyt Yilmaz

Oracle Business Intelligence 11g R1 Cookbook

Credits

About the Author

Cuneyt Yilmaz has been working for Bilginc IT Academy since 2001 as a Senior Consultant and Instructor in Turkey and he's passionate about Oracle technologies. He's delivering training for Oracle and Siebel technologies in around 25 countries in the EMEA region. He mostly specializes in Business Intelligence projects.

He is also a regular speaker in Oracle User Group events in Europe; he delivers presentations about Business Intelligence.

I'd like to express my appreciation to my family, my friends, and my colleagues for their great support in writing this book. I'm also grateful to all who gave me the opportunity to write this book.

I hope you enjoy the book and find it a satisfying source.

About the Reviewers

Vikas Agrawal is a Business Intelligence evangelist with over 15 years of experience working in multiple industries with clients that include Fortune 500 companies. He has deep expertise and knowledge in the areas of Security, Performance, Enterprise BI Architectures, Oracle BI Applications, OBIEE, Informatica, Oracle Data Integrator, Enterprise Data Warehousing, Master Data Management, and Big Data related technologies.

He has been involved with OBIEE from its early days since Siebel's acquisition of nQuire and now as part of Oracle's line of BI tools. He currently leads the BI practice for a prime Oracle partner and has led implementations that have won the prestigious Titan Award for deployment of OBIEE and Oracle BI Applications. He is also responsible for developing product offerings and building the Oracle BI Applications deployment methodology.

In his spare time, he enjoys learning new technologies, ever-changing social media, and marketing use cases, writing new software that helps customers leverage the most out of their investments, travelling, and spending time with his family.

Paul S. Bere is an Oracle Certified Implementation Specialist with over 6 years of experience in OBIEE providing end-to-end business intelligence solutions. His expertise involves working with Business Users, Controllers, Directors, Senior Managers, Executives, Department Heads, and Project Stake Holders. He holds a Bachelor's Degree in Computer Information Technology from Northern Kentucky University.

He is a versatile OBIEE consultant and a computer expert with a myriad of other skills and talents, which include being a soccer referee, a radio host, and a poet. His versatility stems from the development of self-confidence and determination coupled with the grooming of character, constantly refined by good manners, patience, and self-control. He not only believes in intellectual power but also in abiding by the fundamental morals and values of life. He values the discipline to reason sensibly and judge impartially while understanding and respecting others.

Paul is a published author of a poetry book, *The Chronicles of Benevolent Affection: A Declaration of Love, Tribulation and Hope*.

> I would like to thank the author for giving me an opportunity to review this great OBIEE recipe book. I also extend my gratitude and my appreciation to Apollo IT Partners, family, and friends, for their encouragement and support and above all I thank the Almighty Lord for his guidance.

Raunak T. Jhawar graduated as a computer science engineer from the University of Pune (Class of 2009). He has over 3 years of working experience with different organizations and clients small, medium, and large. He works as a data warehouse engineer and specializes in BI solutions for ETL, Data reporting, OLAP, and data mining.

Raunak has reviewed two books that were already published by Packt Publications (Microsoft SQL Server 2012 Professional Tips and Tricks, Microsoft SQL Server 2012 Security Cookbook). He often blogs at `www.sqlservercentral.com` and `www.sqlservergeeks.com`.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- ▸ Fully searchable across every book published by Packt
- ▸ Copy and paste, print and bookmark content
- ▸ On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Instant Updates on New Packt Books

Get notified! Find out when new books are published by following `@PacktEnterprise` on Twitter, or the *Packt*.

Table of Contents

Preface

Organizations store their business-related transactional data in databases or in other various data sources. These data sources are often called Online Transactional Databases (OLTP) and they are designed to improve the performance of business applications such as Enterprise Resource Planning applications and Customer Relationship Manager applications.

This raw data is very important for daily operations, but on the other hand, there is a need for valuable information and knowledge. Obviously, the raw data that is stored in transactional databases needs to be converted to valuable information and knowledge. Business users need analytical reports to make effective decisions.

Business intelligence is a group of processes and technologies that transform the transactional data into valuable knowledge. This enables business users to make correct decisions and thus improves the productivity of their enterprises, especially in the markets where there is huge competition.

Business intelligence is evolving as the time passes and new business challenges emerge every now and then. In past days, business intelligence was related to only historical data. But now, real-time reporting is one of the most important requirements. Before, there were many reporting tools that were part of transactional applications. They were used to generate operational reports. But now, as you will imagine, there are many applications that are based on different technologies in an enterprise and a unified reporting solution is an important requirement.

Oracle had acquired Siebel Systems in 2005. After the acquisition, Oracle introduced a new product named Oracle Business Intelligence 10*g* that was formerly known as Siebel Analytics. The latest version of this product is Oracle Business Intelligence Enterprise Edition 11*g*. OBIEE 11*g* provides every solution to all business requirements. We're going to discuss the features of this product in this book.

What this book covers

Chapter 1, *Exploring and Building the Repository*, discusses the major components of Oracle Business Intelligence Enterprise Edition 11*g* and the basics of the repository. This chapter covers how to create a blank repository and set up the three layers of the repository from the beginning. At the end of this chapter, we're going to upload the new repository and make tests by running sample analyses.

Chapter 2, *Working with Logical Dimensions*, covers the different types of dimension hierarchies in depth. Both the level-based and the parent-child hierarchies will be discussed and you're going to learn how to create them. We will also create the level-based measures and presentation hierarchies.

Chapter 3, *Using Aggregates and the Time Series Functions*, covers the creation and usage of aggregate tables in order to improve query performance. There are two methods of implementing the aggregate tables and both of them will be covered. Also, we will discuss the advantages of the time series functions in this chapter. You're going to learn how to create the measure columns that include these functions in their formula.

Chapter 4, *Working with Multidimensional Data Sources*, offers an overview of the multidimensional sources, which are definitely important in business intelligence projects. We will discuss the implementation of the cubes in the repository. Essbase cubes are going to be used in our sample scenario.

Chapter 5, *Security in Oracle BI*, discusses security concepts. Authentication and authorization methods are also covered. Setting up the permissions on the repository objects, configuring the query limits, and creating the data filters are the other subjects that you'll find in this chapter.

Chapter 6, *Managing Usage Tracking and Enabling the Cache*, covers how to enable usage tracking, as monitoring the users' behaviors is very important in BI projects. Also, we will discuss the advantages of the cache mechanism in the BI server and you will learn how to maintain the cache.

Chapter 7, *Creating Simple Oracle Business Intelligence Analyses*, tells you about the basics of analyses. We're going to construct a simple analysis and discuss the properties of the views that are used in the analyses. You will learn how to change the formatting options of table views and discover the usage of filter types. Also, we will use selections in this chapter.

Chapter 8, *Adding Views to Analyses and Advanced Features*, offers the usage of additional views other than the table view. Pivot table views, gauge views, graph views, column selector views, and the other view types are going to be covered. You will also learn how to configure the master-detail setting in this chapter.

Chapter 9, *Measuring Performance with Key Performance Indicators*, discusses the need for Key Performance Indicators. You will learn how to create KPIs and how to publish them in the dashboards. KPIs are the building blocks of Enterprise Performance Management so scorecards are also going to be covered in this chapter.

Chapter 10, *Creating and Configuring Dashboards*, covers the creation and the configuration of the dashboards in Presentation Services. You will explore the object types that you can publish in the dashboards. You will also discover the dashboard prompts and learn how to create them with all the details.

Chapter 11, *Oracle BI Best Practices*, offers the best practices of the implementation steps in Oracle BI projects. You will find recommendations about the repository including all three layers. Then we will discuss the design of the analyses and the dashboards. Also you will find important tips about performance and security.

Appendix, *The Major Components of OBIEE 11g*, explores the major components of OBIEE 11*g* and discusses the role of each component. Also, processing steps of an analysis will be covered.

What you need for this book

You need to have the following:

- ▸ Oracle Database 11*g* R2.
- ▸ Oracle Sample Schemas (HR and SH): Sample schemas can be installed during database creation or can be installed manually after the database is created. SH Schema could be used for the activities.
- ▸ Oracle Business Intelligence Foundation Suite 11*g*.
- ▸ Oracle Business Intelligence Enterprise Edition 11*g*.
- ▸ Oracle BI Publisher 11*g*.
- ▸ Oracle Essbase.
- ▸ Oracle Scorecard and Strategy Management.
- ▸ A web browser (Internet Explorer 8.0 or above / Mozilla Firefox 3.6 or above).

Who this book is for

This book is designed for the following audience:

- ▸ Business analysts
- ▸ Technical consultants
- ▸ Business Intelligence developers
- ▸ Business Intelligence administrators

It is recommended to know the basics of data warehouses before starting to read this book. Although you don't need to write SQL statements while constructing analyses, it is recommended to know the structure of the SQL statements that will be generated automatically.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "In order to access the variables from the repository, we'll have to use the `VALUEOF` function."

A block of code is set as follows:

```
select distinct 0 as c1,
  D1.c2 as c2,
  D1.c1 as c3
from
SAWITH0 D1
  Order by c2
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "You can easily set the allowed or disallowed periods by clicking on the **Allow** and **Disallow** buttons."

> Warnings or important notes appear in a box like this.

> Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Exploring and Building the Repository

In this chapter, we will cover:

- ▶ Building the Physical layer in the repository
- ▶ Building the Business Model and Mapping layer
- ▶ Adding multiple sources to the logical table source
- ▶ Adding calculations to the fact table
- ▶ Creating variables
- ▶ Building the Presentation layer
- ▶ Validating the repository using Consistency Check Manager
- ▶ Uploading the repository

Introduction

Oracle Business Intelligence Enterprise Edition Suite is a comprehensive reporting tool. The components of the suite are as follows:

Presentation Services

This comprises the presentation of the business intelligence data to the clients through web browsers. This process communicates with the BI server component directly and consists of query clients such as Analysis Editor and dashboards. End users will be able to create and modify analyses or just access business data. According to the business requirements, customized analyses can be created and saved into the Presentation Catalog, which is a repository of Presentation Services. Then we can easily publish these reports using a dashboard.

Oracle BI Server

Oracle BI Server is the main component in the suite. The BI server is simply a query engine that converts the logical requests to a physical SQL statement in order to execute it in the data sources. These optimized queries are generated based on the business rules that are defined in the BI server repository. These logical requests can be triggered from various applications. Obviously the most common one is Presentation Services, which belongs to the OBIEE components group. End users will be able to easily execute the logical requests from the dashboards. One single logical request can be used to query multiple physical data sources. Oracle BI Server and its metadata are transparent to the end users or to the person who executes a logical request. This conversion will be done based on metadata that should be configured by the BI developer. All of the business rules should be created as metadata.

BI Scheduler

This component manages the scheduled tasks. We are going to use either the Presentation Services or the **Job Manager** tool for creating scheduled tasks.

The metadata of the BI server is stored in the repository file on the server where the BI server service is running. A tool named **BI Administration Tool** that is installed with the default installation of OBIEE manages this repository.

In this chapter we're going to create this repository file from the beginning. Having a well-designed repository is crucial in business intelligence projects. We're going to use the Oracle database as a sample data warehouse.

The following are some samples of the data sources we'll also be using:

- ▶ Relational databases
- ▶ Multidimensional sources
- ▶ Flat, XML, and CSV files

The repository is divided into three layers of metadata and are referred to as the following layers:

- ▸ Physical layer
- ▸ Business Model and Mapping layer
- ▸ Presentation layer

We're going to start with the Physical layer.

# Building the Physical layer in the repository

The Physical layer defines the physical data sources and the Oracle BI Server uses these definitions in order to submit queries. The Physical layer is the first layer that we have to create. There could be many data sources based on different technologies and their metadata may exist in this layer.

How to do it...

1. We're going to use **BI Administration Tool** to create the repository and specify the business rules and object definitions. After we open **BI Administration Tool**, we are going to create a new repository from the **File** menu and save this new repository to its default location.
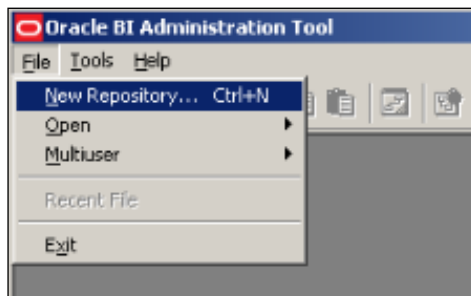
2. We'll have to specify the name of the new repository and its path. We can easily import the metadata with the wizard or skip importing the metadata at this moment. We're going to start the import process later on. So entering the password in **Repository Password** will be enough.
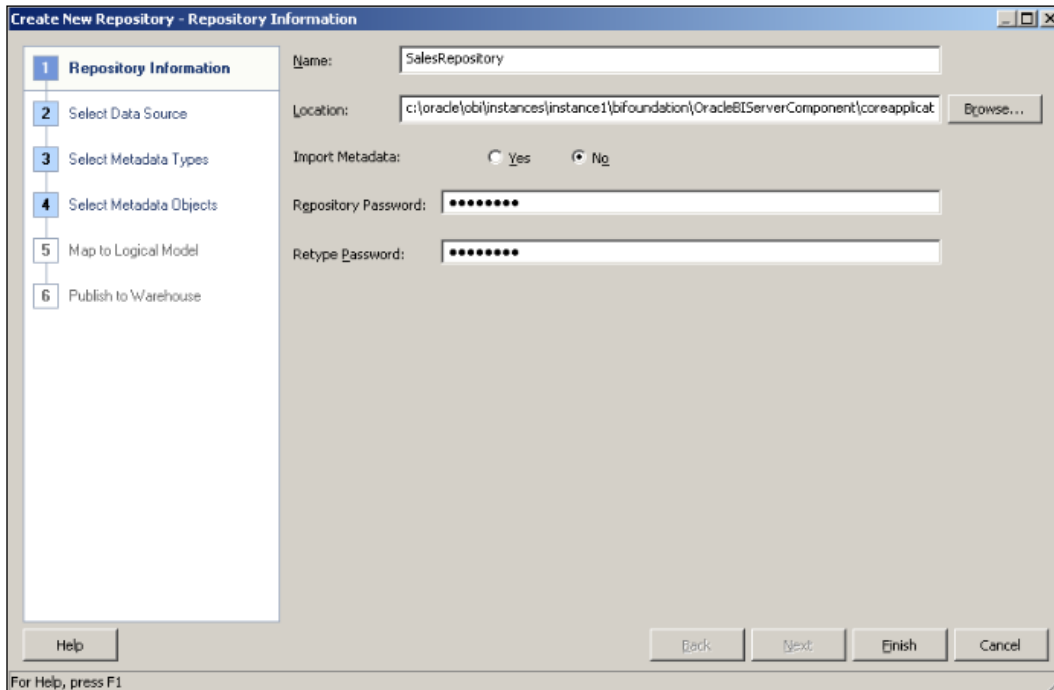
3. Clicking on **No** in the **Import Metadata** option will prompt the wizard to finish. Then it's going to show the content of the repository. We're going to see only three layers without any object definition.
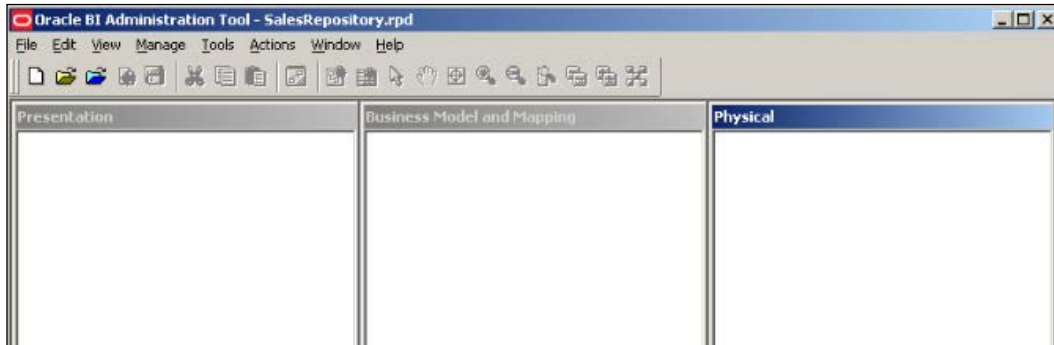
4. These values should be entered in the **Import Metadata - Select Data Source** window:

   ❑ **Connection Type**: The type of the data source should be defined in this field. One of the technologies should be selected from the drop-down list. We're going to use Oracle Database 11*g* R2 Version in our scenario, so the type is `OCI 10g/11g` for Oracle.

   ❑ **Data Source Name**: This is the name that is configured in the `tnsnames.ora` file. By default, this file is located in the `$ORACLE_HOME/network/admin` directory.

   ❑ **User Name** and **Password**: The last parameters are the username and the password of the user who has access to the tables.
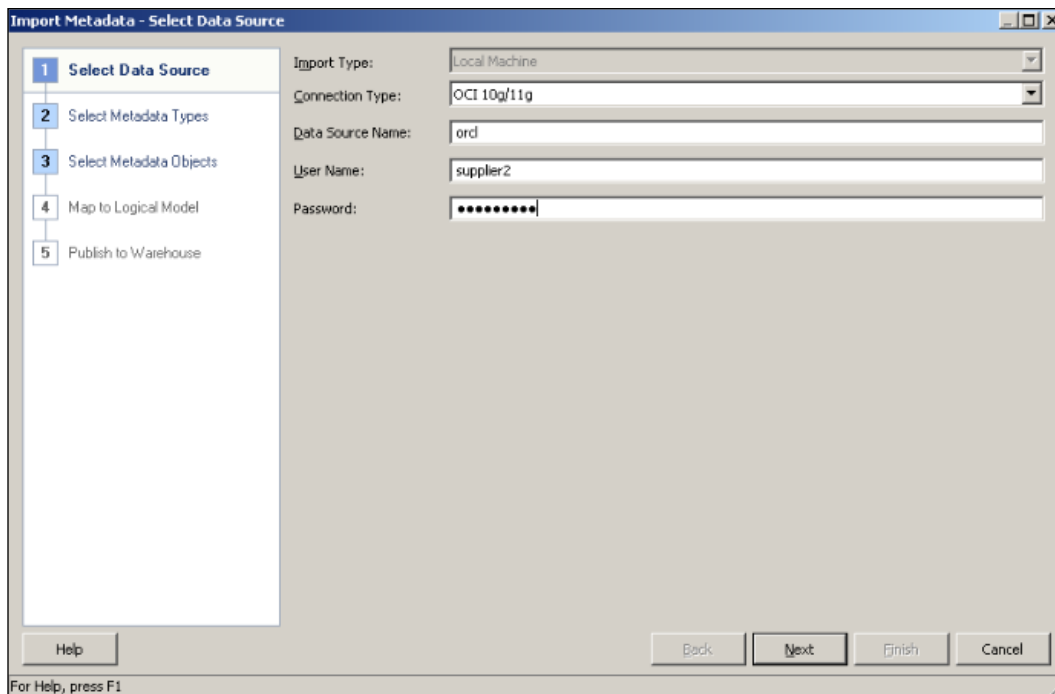
5. The type of the metadata objects should be selected in the **Select Metadata Types** window. Then we're going to select the tables that we're interested in, in our BI project. So only four tables from `Supplier2` schema are selected. We can easily import new tables when needed. Additionally, constraint definitions are also retrieved during the metadata import process.
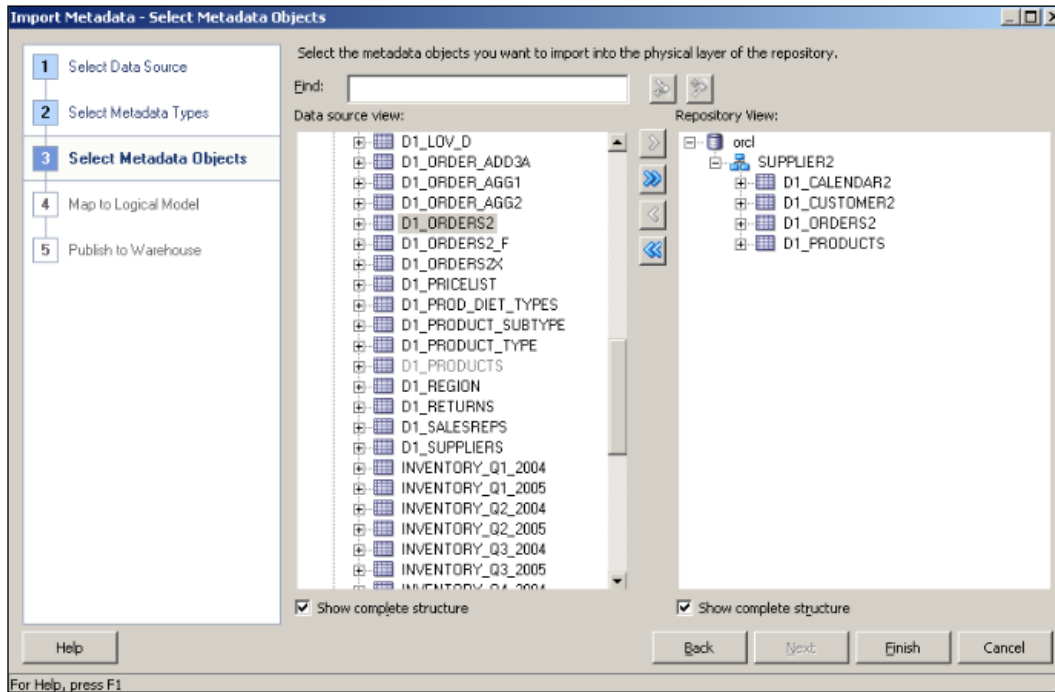

6. After the wizard ends, we'll see that the database object definition is created and one Connection Pool object is created as well. It's the definition of the connection to the physical data source. We'll also notice that four of the table definitions are imported under the schema of `Supplier2`. Now alias object definitions are going to be created. They are the objects representing the physical tables. Referencing to the same table twice in the same SQL statement causes circularity, and this situation is not allowed in the OBI repository. In order to avoid circular joins, alias object definitions are needed. Right after the import wizard ends, alias object definitions should be created by the developer manually. We'll right-click on the table and navigate to **New Object** | **Alias** from the menu list.
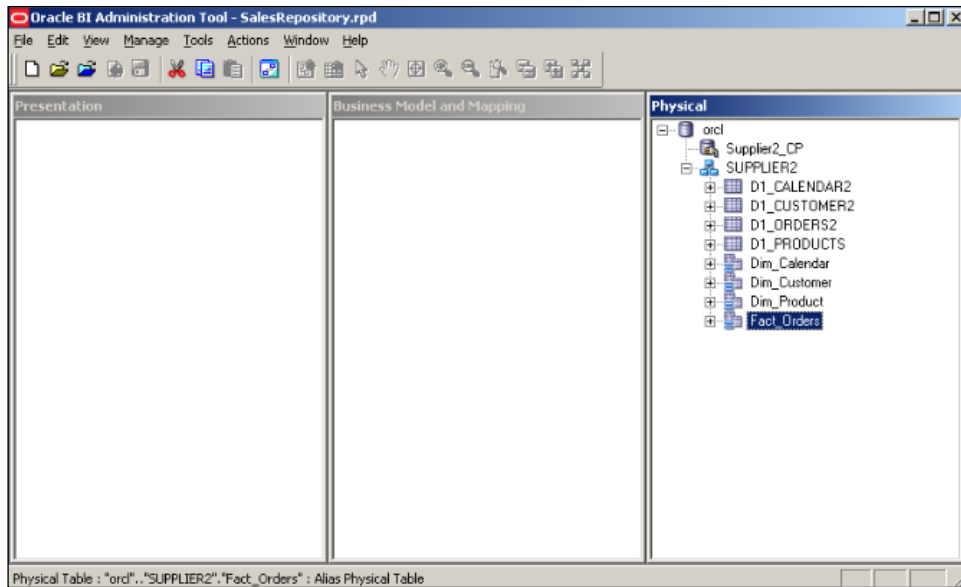
7. Actually, there are not any foreign key constraints defined on the tables in the database. In these kinds of cases, the primary key and foreign key relationship should be defined in the BI repository by creating joins. In order to create joins, select all aliases and right-click on them to open the **Physical Diagram**. Right after it, we'll use the **New Join** button to create the joins between the tables. It's going to ask about the columns that will be used in the join definition.
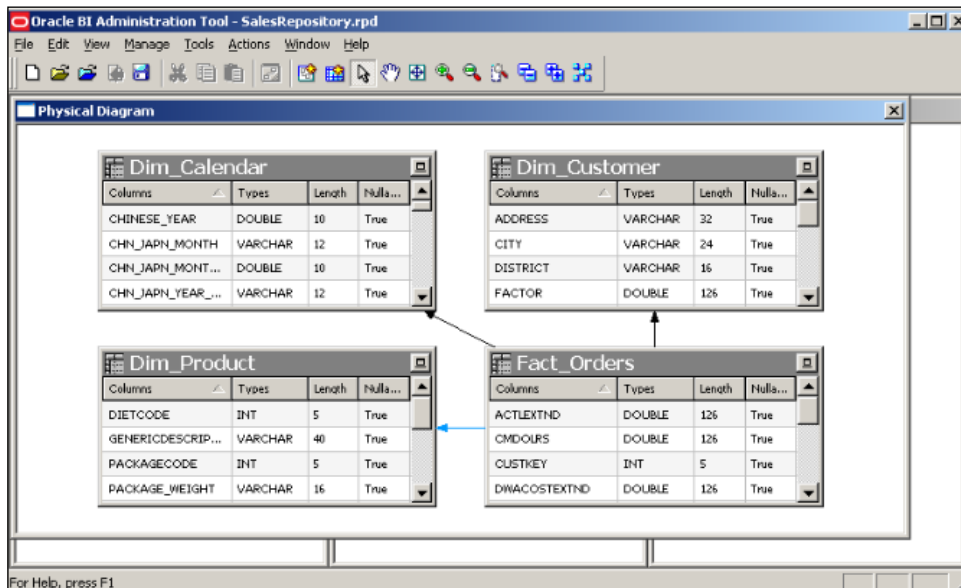
8. The definition of each join object looks like the one in the following screenshot. There's no information about the type of join in the Physical layer. So the join type will be set in the Business Model and Mapping layer. We should only specify the column mappings from the fact table to the dimension tables one by one. The foreign key columns in the fact tables will be referencing to the primary key columns in the dimension tables.
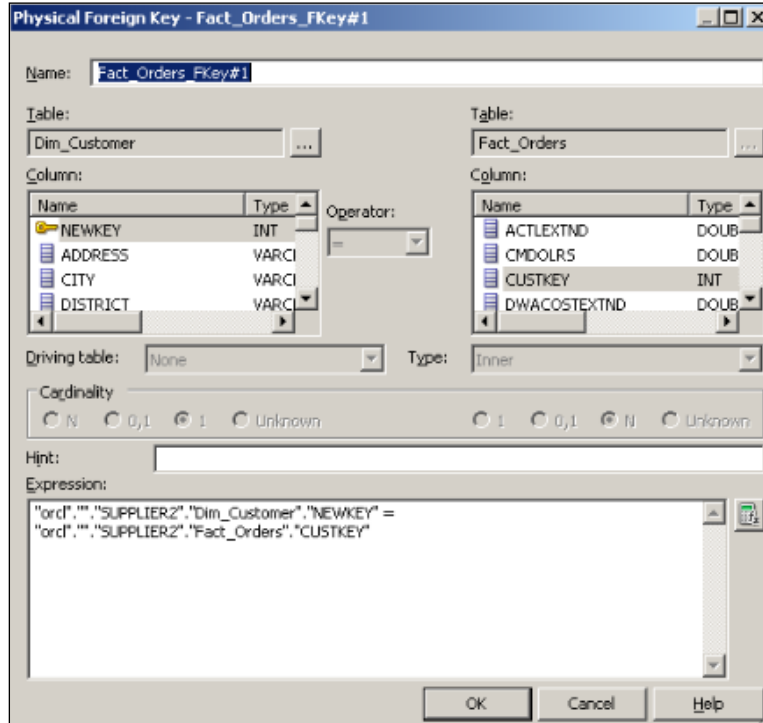
How it works...

Now we've finished the basic tasks of the Physical layer. Physical tables and their aliases are created and also all the relationships between the tables are defined. We can say that creating the Physical layer is easier than the BMM layer. Most of the tasks are done by using wizards that take less time. The objects that are in the Business Model and Mapping layer are going to reference to the Physical layer objects. Obviously, having a well-designed Physical layer will make the other tasks in the BMM layer easier. When any report is executed at Presentation Services, the logical SQL statement is going to be generated based on the definitions in the BMM layer. Then this logical SQL statement is going to be converted to a physical SQL statement that is going to be executed at the database.

There's more...

We only used some tables in our sample scenario. Actually, the Physical layer objects are dependent on the data sources. Let's imagine that if multiple databases, Excel spreadsheets, XML files, or multidimensional data sources exist, then all of them should be defined in the Physical layer. We're going to focus on multidimensional data sources in *Chapter 4*, *Working with Multidimensional Data Sources*.

Building the Business Model and Mapping layer

The Business Model and Mapping layer defines the logical model of the data that maps the business rules to the Physical layer objects. There can be many business models that can be mapped with different kinds of data sources based on different technologies. Also dimensional modeling should be done in this layer. This can be achieved by creating dimension objects and their hierarchies. Also all about the calculation and aggregation rules will be defined in this layer.

As a summary, all the business rules are going to be created in this layer. It's probably the most important layer in the repository, and it'll definitely take longer to create the objects in this layer.

Creating business models can be done by two methods, manually or by drag-and-drop. If the manual object creation method is preferred, we'll have to create objects in the following order:

1. Create the business model.
2. Create the logical tables and map them to the physical tables.
3. Create the logical columns.
4. Define the logical joins.

How to do it...

1. Create the business model manually by right-clicking on the BMM layer and selecting **New Business Model**. (Give it a name, in this case `Sales`.)

2. Ensure that the **Disabled** option is checked because there's no other object in this business model. We're going to change this setting later.
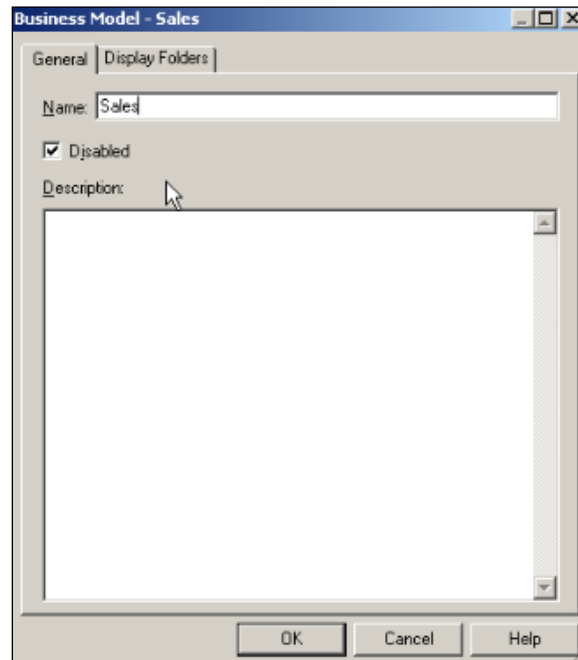

3. After creating a blank and disabled business model, drag-and-drop the physical objects from the Physical layer onto the business model named `Sales`.

   Automatically created objects are:

   - Logical dimension and fact tables
   - Logical columns and their mappings to Physical layer
   - Logical table sources for the dimension and fact tables
   - Logical keys and joins (can be accessed from the **Business Model Diagram** window)

4. Now it'll be good to make a clean-up because with this automated process all of the columns from the Physical layer are created as logical columns. Most probably we won't need all of the logical columns; we can easily delete the logical columns that are not required by right-clicking on the columns. Another task is renaming the logical tables and columns with meaningful names.
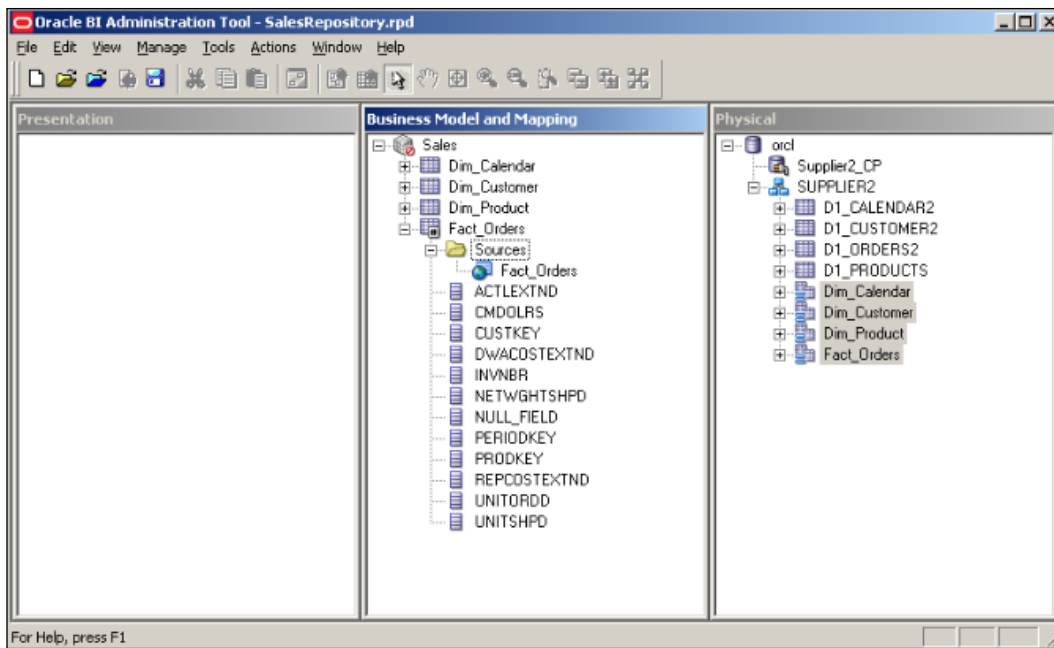
5. After making the clean-up, we'll see that some columns are deleted and some are renamed.
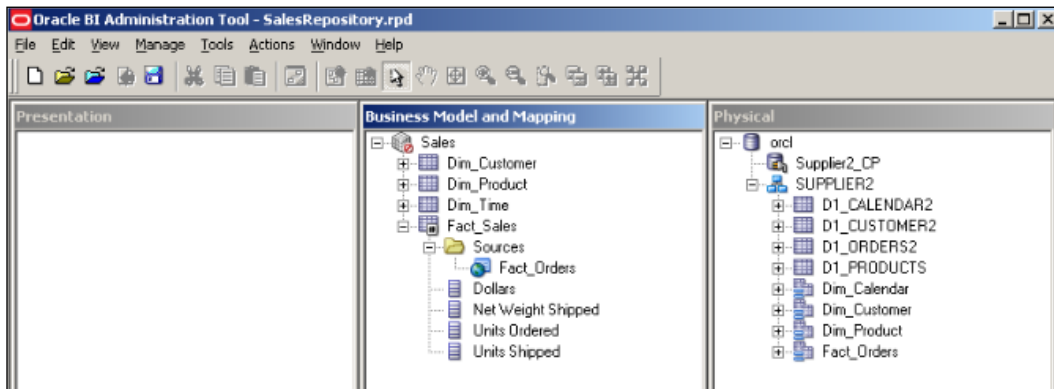
6. When we open the properties of the logical table source named `Fact_Orders`, we'll find mappings from the **Logical Column** to the **Physical column**. Also if you need to access other details of the logical table source, you'll have to extend the logical table sources folder, so you'll find the logical table source definition.
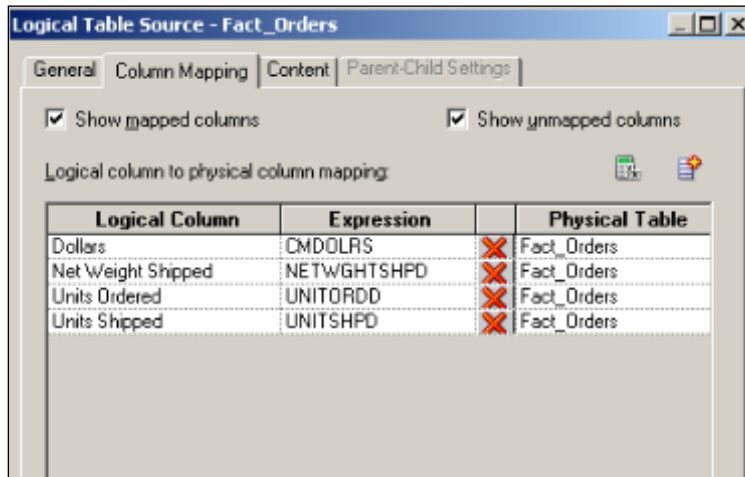
7. The new business model is almost ready but still there is a missing step that is the aggregation rule for the columns in the fact table. By default, it's not set. The following are the options for the aggregation rule, so depending on the requirement you can use one of them:

   ❑ `Sum` and `Avg`

   ❑ `Count` and `Count Distinct`

   ❑ `Max` and `Min`

   ❑ `StdDev` and so on

8. In order to set the aggregation rule for one logical column in the BMM layer, you'll have to open the properties window of that column and go to the **Aggregation** tab. You'll find the rules in the drop-down list and select one of them. The most common one will be `Sum`. So we'll use `Sum` as the aggregation rule in this demonstration. In the case of business requirements, we can use different aggregation rules. If you need to display the number of the rows, we can use `Count` as the aggregation rule.
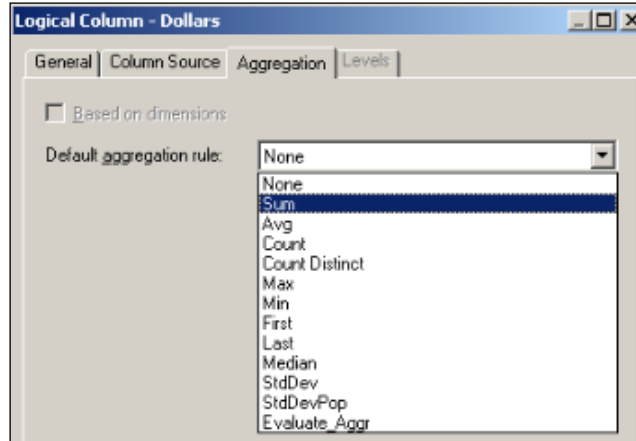
9. So we've configured all of the measures in the fact table with the `Sum` aggregation rule.
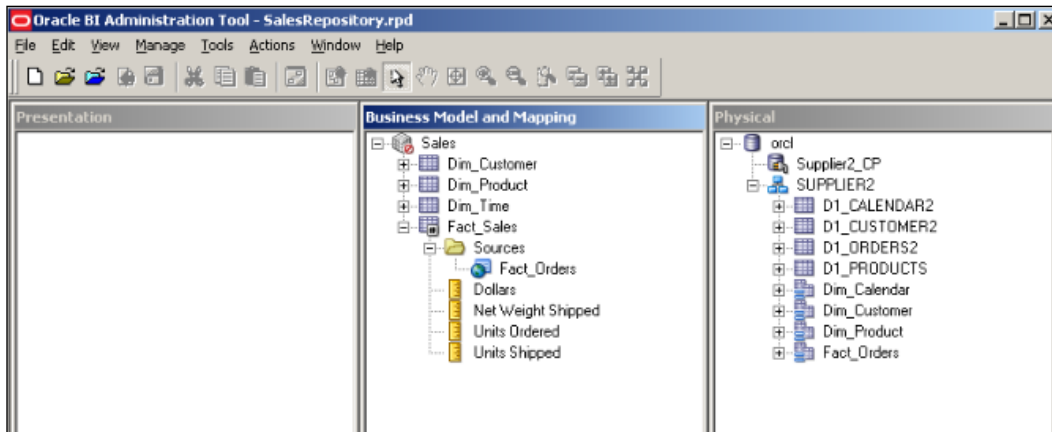
How it works...

We created a simple business model in this section. Each logical table should have at least one logical table source. We might create multiple logical table sources as well, just in order to use summary tables or horizontal/vertical data federation. The important point in this layer is creating the logical joins. The Oracle BI Server will recognize the logical fact and dimension tables by checking the logical joins. Obviously, the logical joins are dependent on the physical joins, but we may have only one table in the Physical layer. So there won't be any physical join. Even in a single table scenario, we'll need to create fact and dimension tables in the BMM layer and create logical joins between them. At the end, all the objects will be referencing to only one single table. After working on the Business Model Diagram, we selected the aggregation rules that will be applied on the measure columns such as `Sum`, `Count`, and `Avg`. Is this a required step? Actually, it's not required. If you don't set the aggregation rule of the measures, you won't be able to see aggregated data in the reports. Let's assume that if the fact table contains one million rows, you'll see one million rows in the report. Although setting the aggregation rule is not a must, it's highly recommended to do it.

There's more...

Regarding the business model, there is one more object type called logical dimensions. The Drill Down feature and aggregate (summary) tables can be used after dimensions are created. As you see, two important features are dependent on dimensions. We're going to focus on logical dimensions and their types in *Chapter 2*, *Working with Logical Dimensions*.

Adding multiple sources to the logical table source

Because of the data warehouse design, data may be stored in the physical tables in the normalized way. That means some columns are stored in different tables. This kind of design is called as the **snowflake** schema. For example, let's assume that the product data is spread into different physical tables, for example, the price information doesn't exist in the product table and it's stored in the price list table. Another example is that supplier data can be stored in another table such as the suppliers table.

When this is the case, there's a business solution in Oracle Business Intelligence. We can add multiple sources to existing logical table sources.

How to do it...

1. In order to add multiple sources, we'll need to extend the existing repository and import these additional tables to the Physical layer. We're going to avoid using the physical tables directly, so we will create aliases for these new tables. You can see that they are already imported and aliases are created. Additionally, physical joins should be created.
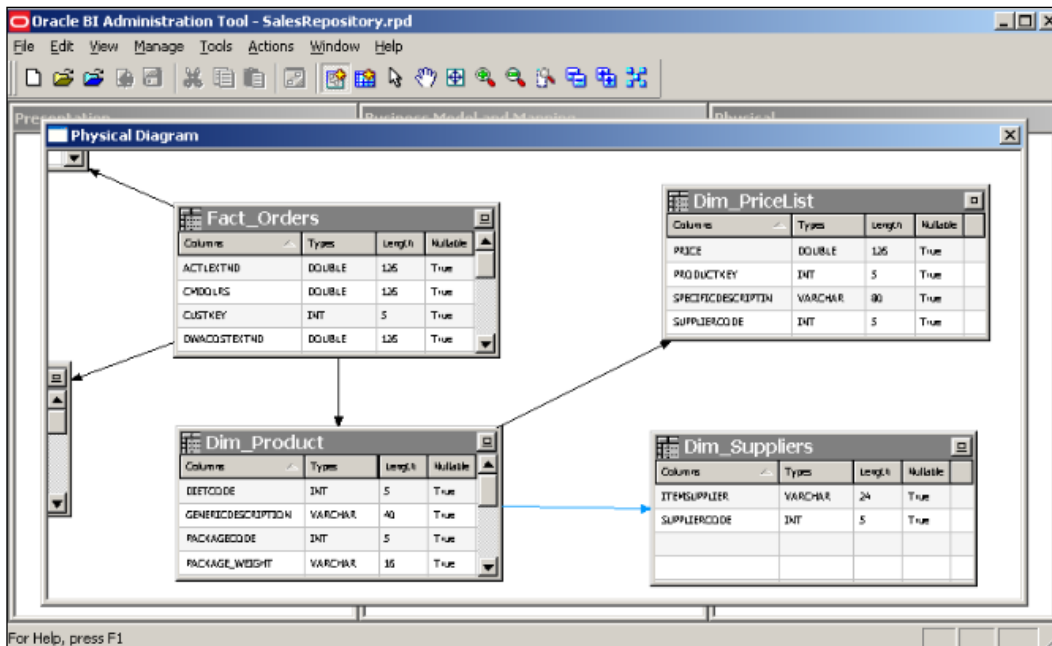

2. Now we have to reflect these changes to the BMM layer. Although the new physical table definitions are created, the BMM layer is not aware of this information. Again there are two methods of making logical changes in the **Business Model and Mapping** layer, manually or drag-and-drop. We're going to drag-and-drop the new physical columns on to the logical table source (not on to the logical table).

When you drag the two columns named `PRICE` and `ITEMSUPPLIER` from the
Physical layer on to the logical table source, you'll see that two new logical
columns are added into the `Dim-Product` logical table.
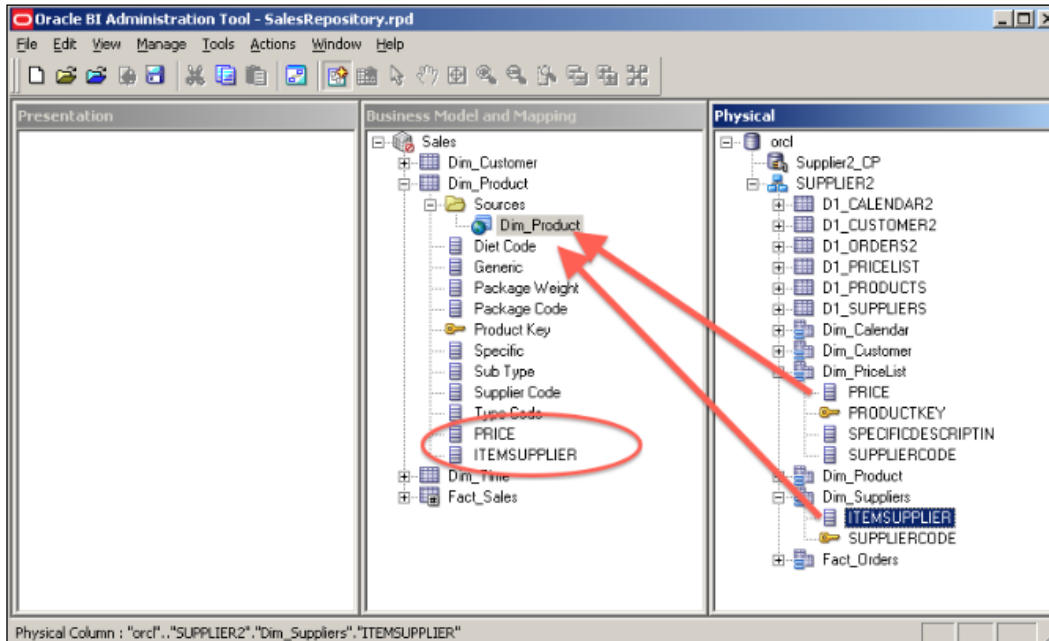
3. After this action, there will be a modification in the logical table source. Normally, the
   logical table sources are mapped to a single physical table. But now you'll notice at
   the end of this action that the `Dim_PriceList` and `Dim_Supplier` physical tables
   are also mapped with that logical table source. By default, it uses the **Inner** join type
   and we can change it based on the requirement analysis.
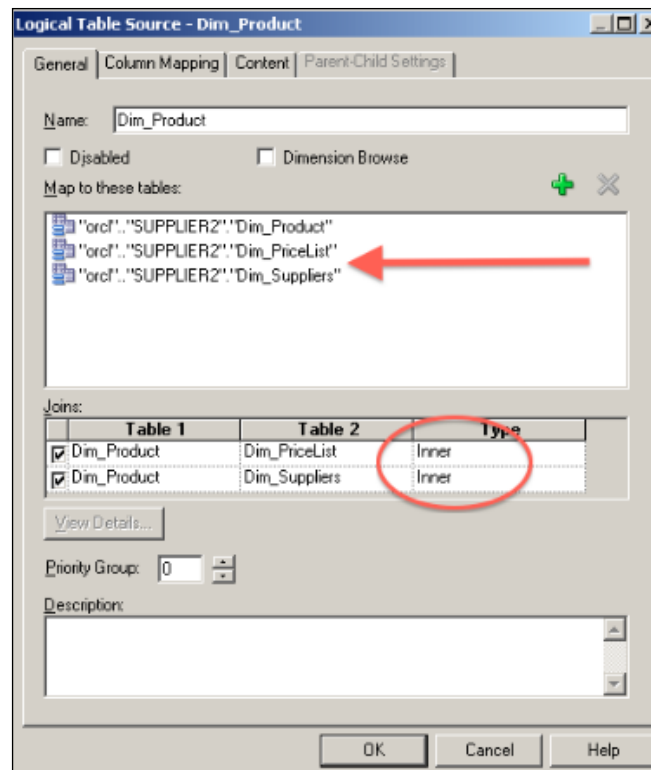
4. Another change in **Logical Table Source** is **Column Mapping.** It also automatically maps the new logical columns with the physical columns.
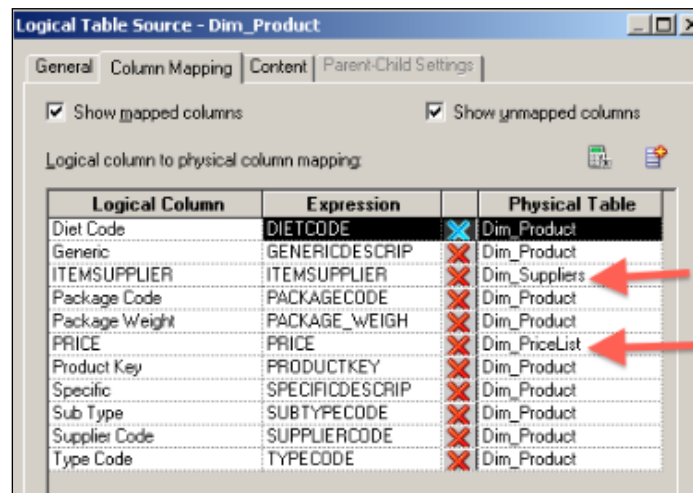
How it works...

We added multiple sources into the logical table sources in this recipe. We could use the manual method, which would cost more time and steps, so we used the drag-and-drop method. If the supplier and price columns are used in the report the BI server is going to retrieve the data from the two different tables. This processing relies on the settings of the column mappings. Whenever a report is executed, the BI server is going to check the logical table sources and the column mappings. All logical columns should be mapped with the physical columns, otherwise the consistency check tool will generate an error.

Designing of the business model is a crucial step. Any mistake at this layer will cause generation of inaccurate result sets. When we start to construct the reports at Presentation Services, the logical SQL statement is going to be generated, based on the business rules in the model. It'll check the logical relationships, column mappings, aggregation rules of the measures, and so on.

There's more...

We only created one business model in our scenario that doesn't contain any dimensions or multiple logical table sources. The business models are referenced by subject areas from the Presentation layer. They focus on a particular business view. We might create more business models according to the business requirements such as a sales business model and a finance business model.

Although we're going to focus on the Presentation layer in another chapter, we shouldn't forget a critical limitation in the tool. Subject areas in the Presentation layer cannot span multiple business models. They cannot be mapped to multiple business models. So if there's a logical table that is going to be needed in all of the business models, it should be copied to all business models.

Adding calculations to the fact table

Business users are going to be interested in some calculations of the values in the measure to compare some values. So at the end, they will need valuable information from the existing fact tables that contain measures. For example, in our case they may be interested in comparing the units ordered with the units actually shipped. The business solution of this scenario is to add new calculations into the logical fact table. There can be another solution such as adding these calculations into the data warehouse and making the calculations at the database level. But this solution will take more time. So that's the reason that we're going to add new calculations into the BMM layer.

There are three types of methods that we can use:

- ▶ Creating calculations based on logical columns
- ▶ Creating calculations based on physical columns
- ▶ Creating calculations using the Calculation Wizard (based on logical columns)

Besides these calculations, we're going to cover time-based calculations in *Chapter 3, Using Aggregates and the Time Series Functions*.

How to do it...

1. First, we're going to use the first method, which is about calculations based on logical column sources. We're going to expand the logical fact table and see the list of existing columns, then right-click on the table and navigate to **New Object** | **Logical Column...**. It's going to open the **Logical Column** properties.
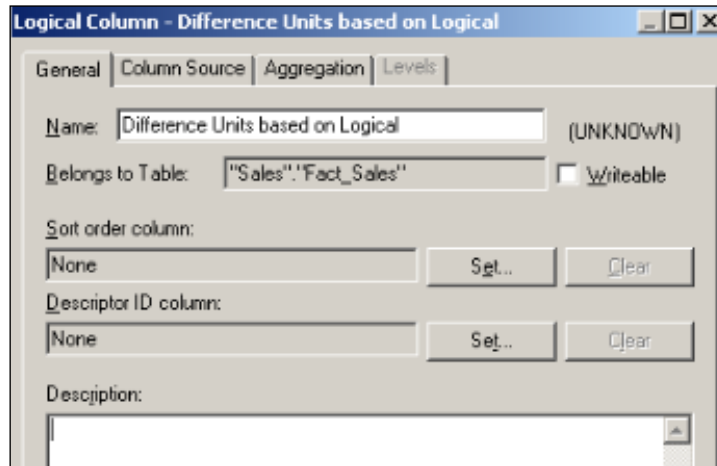
2. By default, this **Logical Column** is not mapped to any physical column. When you click on the **Column Source** tab in the same window, you can see that it's not mapped yet. We're going to select the **Derived from existing columns using an expression** option box from the **Column Source Type** section. You'll see the Edit Expression icon.
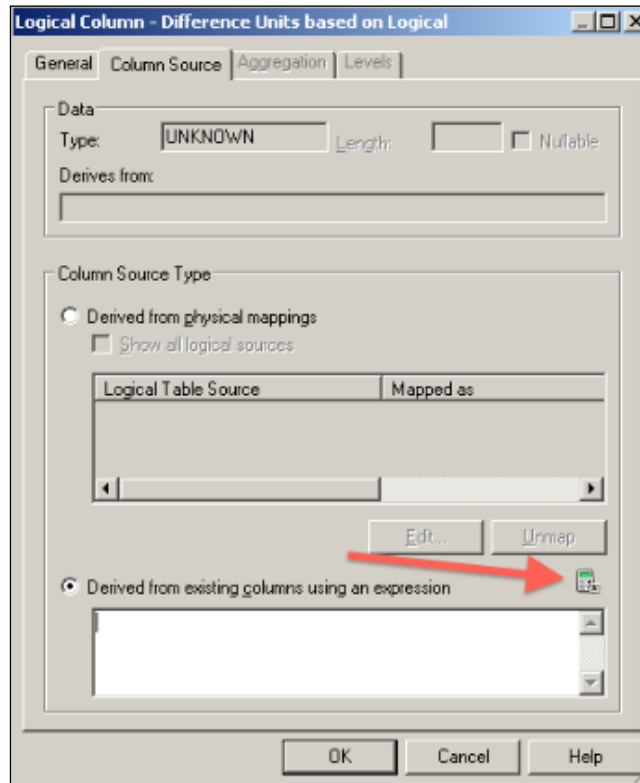
3. When you click on the Edit Expression icon, it's going to open **Expression Builder** where we're going to write the formula and click on the **OK** button.

```
"Sales"."Fact_Sales"."Units Ordered" -
"Sales"."Fact_Sales"."Units Shipped"
```

4. By using this method, we can see the differences between the two measures such as `Units Ordered` and `Units Shipped`. The most important point is how the calculations and aggregations are done in this method. When you use the logical columns in the calculations, it means that the aggregation rule is going to be applied before the calculation. So the formula is as follows:

```
SUM (UNITS ORDERED) - SUM (UNITS SHIPPED)
```

The formula will not be:

```
SUM (UNITS ORDERED – UNITS SHIPPED)
```

5. When it comes to the second method, that is when the calculations will be based on physical columns, we're going to again create the **Logical Column** in the logical fact table. But this time we're not going to change anything in the **Column Source** tab, because we've already discussed that physical column mapping can be only accessed from **Logical Table Source**.
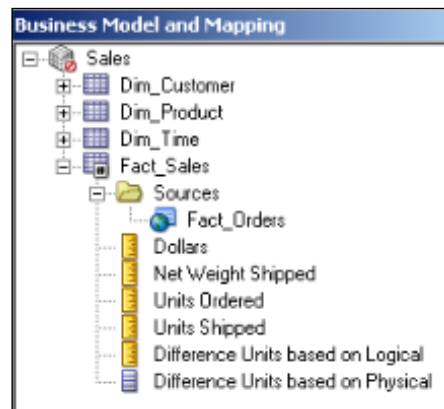
6. And when we open the **Logical Table Source** window, we'll see that the new column is not mapped yet.
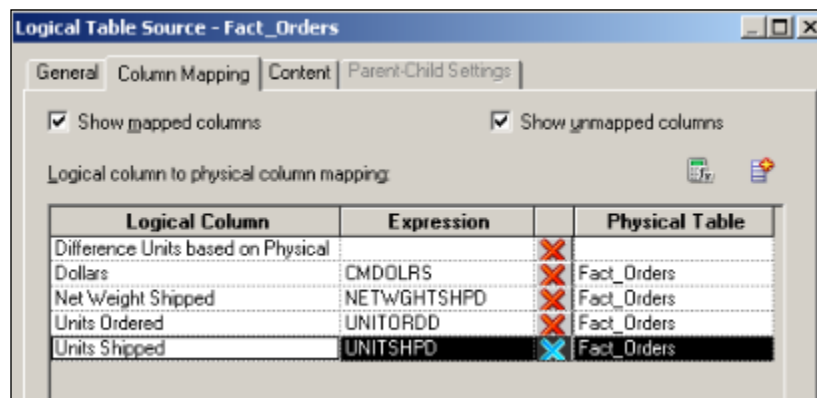
7. Again we're going to open **Expression Builder** and write the formula, but this time physical columns are going to be used in the formula.
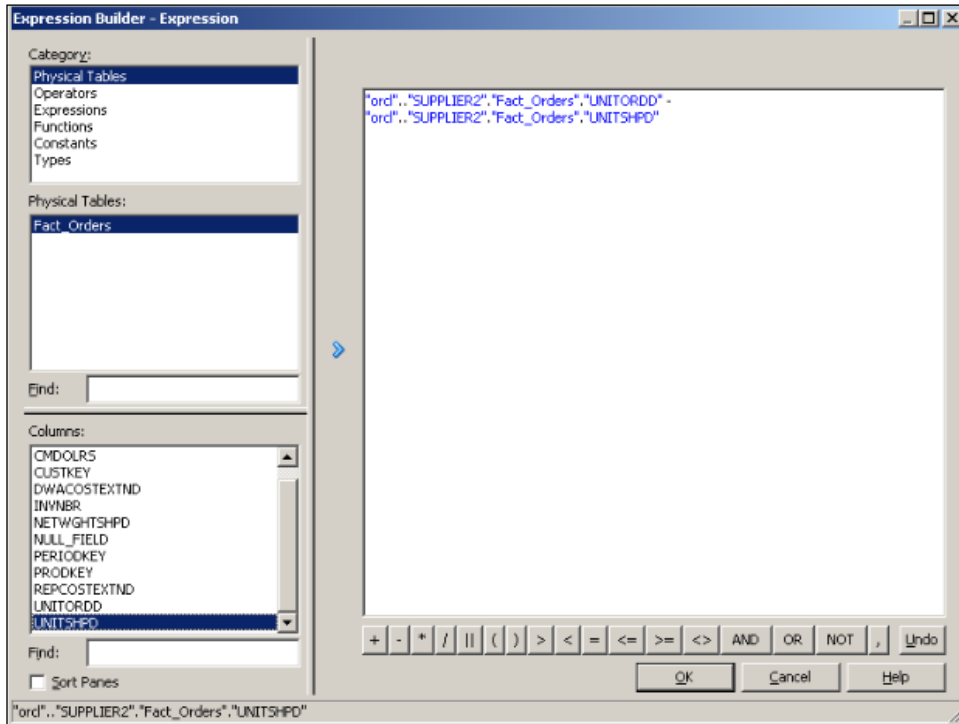
8. As a result you'll have another measure that calculates the difference between the units ordered and the units shipped. Now the aggregation rule is going to be applied after the calculation as shown:

```
SUM (UNITS ORDERED – UNITS SHIPPED)
```

As an extra step, you'll need to configure the aggregation rule that will be used for this new Logical Column. Otherwise, the BI server is not going to be aware of the aggregation rule that is going to be applied.

9. In the previous method, we could also use the wizard. It's easier to use it. You just have to right-click on the logical column that you want to make a calculation in and go to the **Calculation Wizard** menu item; it'll ask you the other values for calculations.
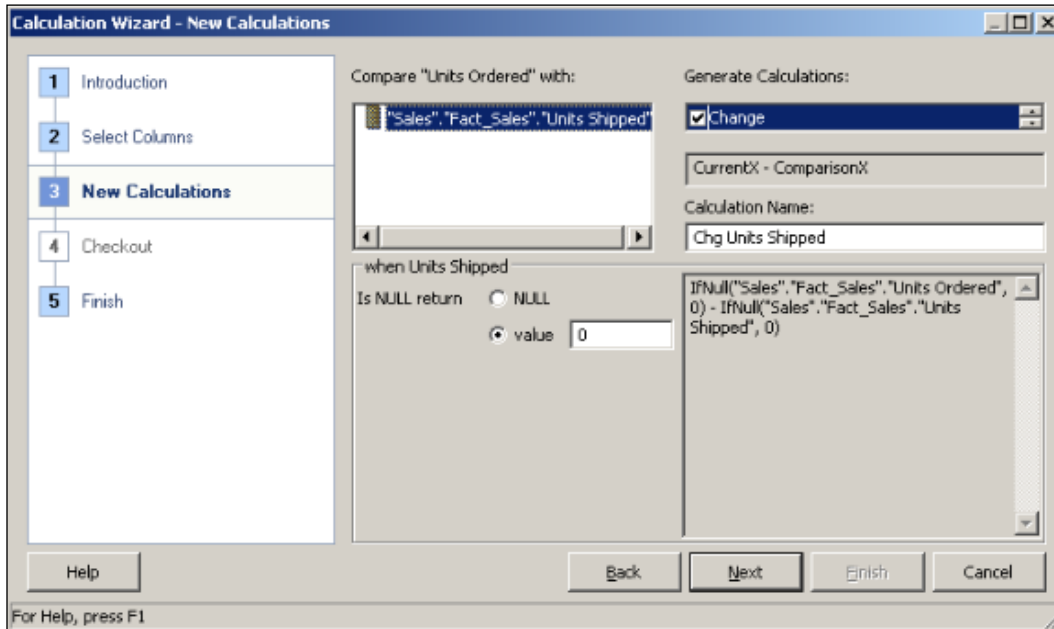
At the end of the wizard you'll notice that the result is exactly the same as the first method, creating calculations based on logical columns, and the aggregation rule is going to be applied before the calculations.

How it works...

We learned how to create new calculations in this section and also covered how the aggregation rules are applied based on the columns. These calculation techniques may produce different results when it comes to the usage of multiplication or division.

There's more...

Although it's possible to add new calculation measures into the logical fact table easily, it's better to implement them in the **data warehouse,** because whenever a query is executed, these new measures are going to be calculated every time during runtime. This is going to negatively impact the query performance. The recommended way is implementing these measures at the database level and handling them in the **Extraction, Transformation, and Loading** (**ETL**) **process**. As a result, when the query is executed, the calculated values will be retrieved from the database itself instead of making any calculation during runtime. Adding new calculations to the business model should be a short-term solution. Obviously creating these calculated measures at the database level has a cost. Data warehouse design and the ETL process are all going to be affected, and it'll take some time to implement them.

Creating variables

One of the important requirements for creating dynamic reports are variables. Variables can store values in the memory and can be accessed by name. We're going to create these variables by using the **Variable Manager** tool that is accessed from the **BI Administration Tool** window. There are two different types of variables:

- ▶ **Repository variables**: They persist in the memory when the BI server service is started and until it's shutdown. They have two forms, **static** and **dynamic**. Static variables are like constants. You cannot change their value. Dynamic repository variable values are set when the BI service is started. Then you can easily change their value. Only one instance is created for repository variables and their values can be set by creating an **initialization block**. These initialization blocks can also be scheduled in order to reset the values.

- ▶ **Session variables**: They persist during a user's session period. There are two types. One of them is called a **system variable** and the other is called a **non-system variable**. Unlike repository variables, session variables have many instances based on the number of the sessions. We can use initialization blocks in order to set their value in the beginning. But unlike repository initialization blocks, these blocks can be rescheduled.

How to do it...

1. Open the **Variable Manager** block from the **Manage** menu.
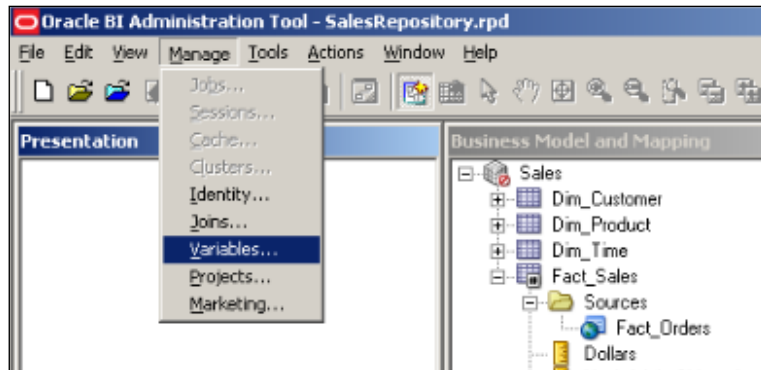
2. When the **Variable Manager** is opened, you'll see that variables are grouped into two main categories, **Repository** and **Session**.
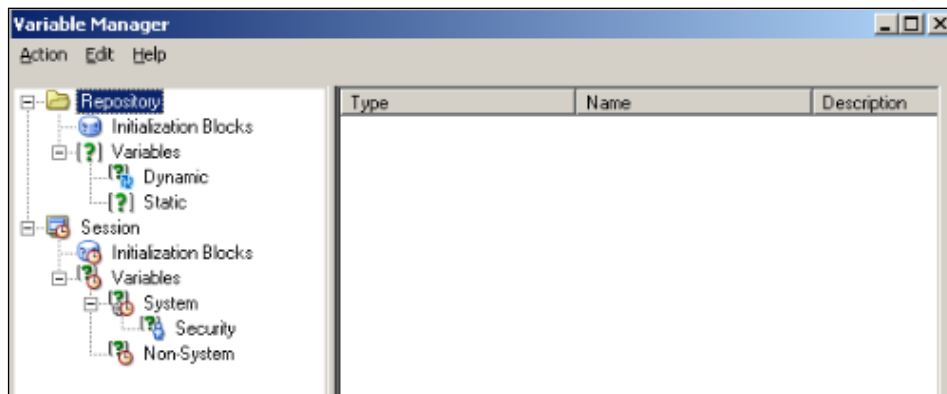
3. Create a repository variable as an example. We're going to click on the **Action** menu and go to **New | Repository variable**, and it's going to open a new variable window. As you'll see there are two options for the type of variable, static and dynamic. Let's assume that we need dynamic variables. So in this window we're going to click on the **New...** button.
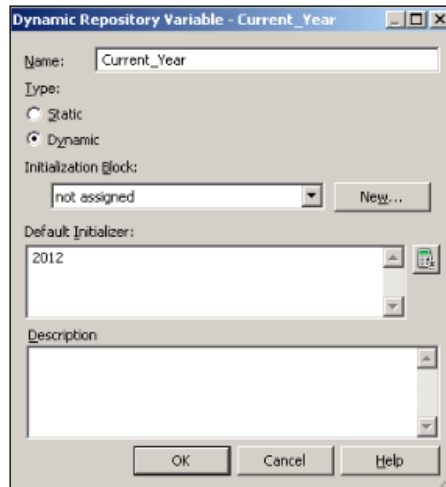
4. This time a new **Initialization Block** window is going to open. First we will schedule this block regarding when it's going to be executed in order to set the variable's value. We're going to use 1 hour as the period.
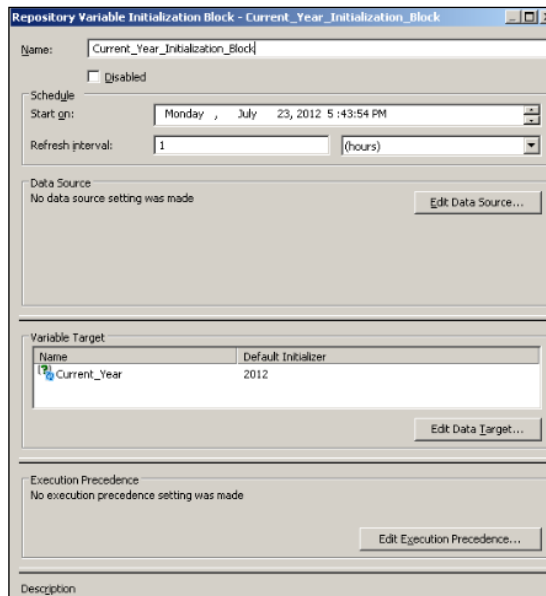
5. The second important configuration is editing the data source. When you click on the **Edit Data Source...** button, you'll have to specify the **Connection Pool** name and the query that you want to execute.
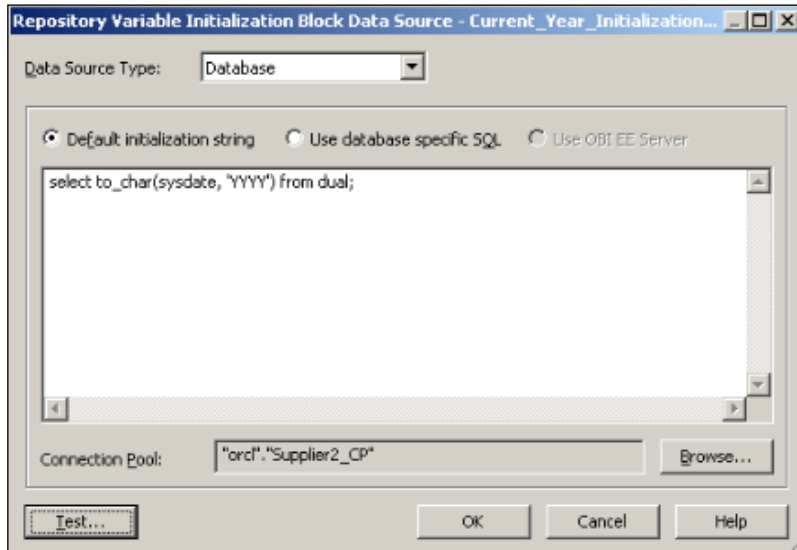
6. You can also test your query by clicking on the **Test...** button. The output will be similar to the following screenshot:
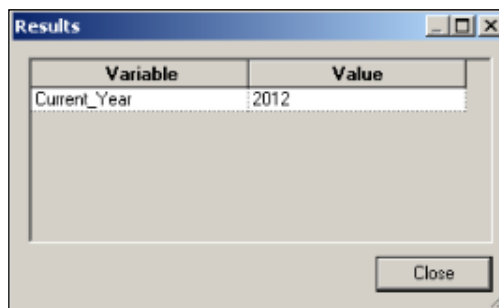
7. At the end, you'll see that both the dynamic repository variable and the initialization block are created. It's also a good practice to create different connection pools for the initialization blocks.
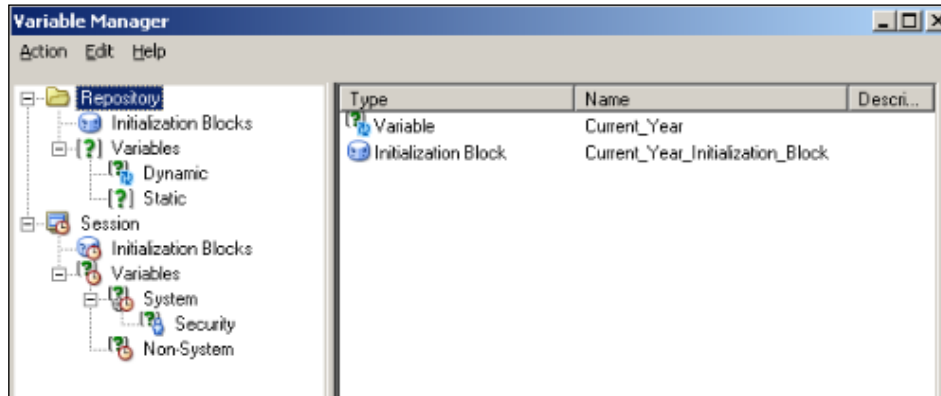
How it works...

We covered the repository variables in this recipe. Variable object definitions are very important from both the end user's and the BI developer's perspectives. When it comes to dynamic reports, we'll need to use them. Business users can access these variables from the Presentation Services. BI developers should share these variables with the business users because we do not have a list of variables on the Presentation Service.

These variables are accessible either from the repository or from the Presentation Services. In order to access the variables from the repository, we'll have to use the VALUEOF function. Let's assume that there's a variable named Current_Year. If you want it to have a dynamic design in the repository, you'll call the variable as follows:

```
VALUEOF("Current_Year")
```

This statement is going to return the value of the Current_Year variable. The VALUEOF function is expecting one argument, that's the name of the variable. Let me remind you again that variable names are case sensitive. When it comes to accessing them in the Presentation Services, we'll just write the name of the variable and we won't use the VALUEOF function.

There's more...

Creation steps of the session variables are very similar. You can create the session variables and then the initialization blocks in the same way that we create the repository variables.

Building the Presentation layer

Now in our basic repository the only missing part is the Presentation layer. The Presentation layer is the last layer that will be built and it exposes the customized view of the existing business models to business users. The Presentation layer provides meaningful data in a simple way. For example, all of the logical columns are not required to be published. Key columns can be removed from the Presentation layer to provide a simpler view. We can reorder the tables and rename the table names or the column names. So it's a representation of the existing business model in a well-designed and simple way.

We're going to create **subject areas** in this layer in order to group the tables based on different views for different business requirements. A single subject area can be mapped to only one business model; it cannot access two business models. So this will be one of the limitations when it comes to designing our project.

How to do it...

1. We can create the object definitions manually or by the drag-and-drop option as in the previous layers. It'll take less time when you compare it with the Business Model and Mapping layer. Now we're going to drag-and-drop the business model from the BMM layer on to the **Presentation** layer. You'll see that one subject area with the same name as the business model is created automatically. Then we can change the order of the tables or order of the columns, or we may rename the table column names.

After creating the first subject area you'll see the relationship between the layers. A presentation table named `Fact_Sales` is mapped with the logical table named `Fact_Sales` and this logical table is mapped with the physical object named `Fact_Orders`. The `Fact_Orders` object, which is obviously not a table, is an alias that is referencing to the physical table named `D1_ORDERS2`.
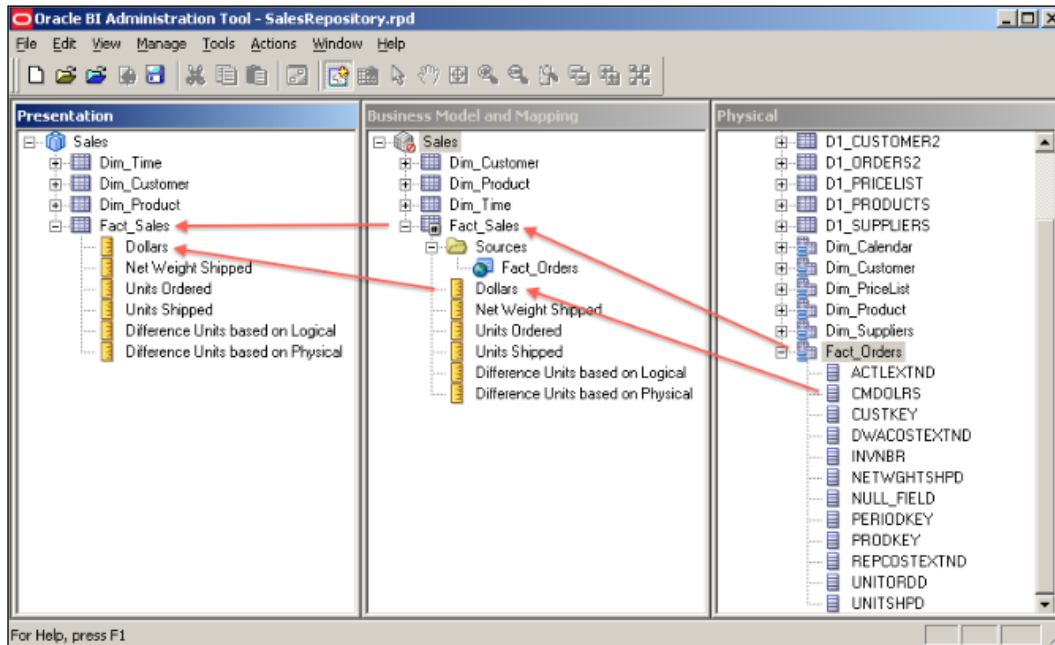
2. When you open the properties of **Subject Area** named **Sales** in the **Presentation** layer, you'll see the **Presentation Tables** tab. You can change the order of the tables from this tab.
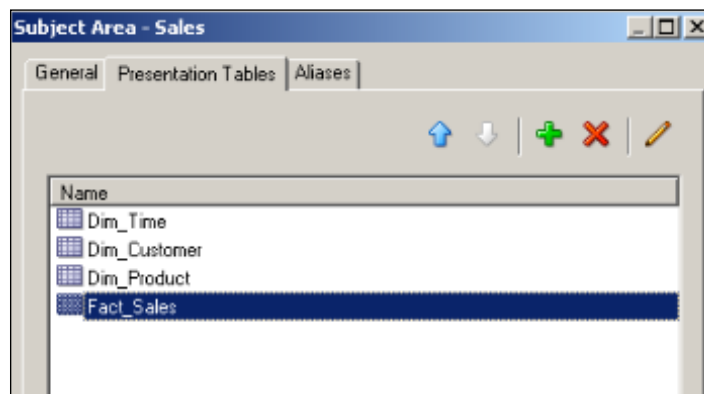
3. In order to change the order of the columns, you'll have to open the properties of **Presentation Table** as shown in the following screenshot and use the arrow keys:
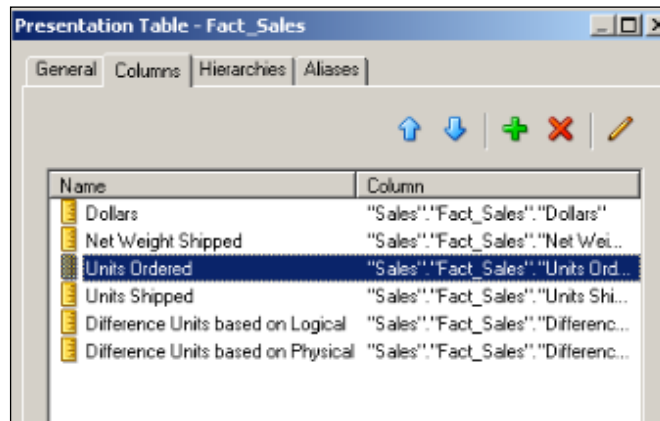
How it works...

Maybe the easiest part of the repository is the Presentation layer, but this doesn't mean that we should not take care of this layer. It's easy as much as it's important, because this is the layer that is published to the end users through the Presentation Services. The names should be meaningful and we'll also have to think about the order of the tables. The list of the presentation tables and presentation columns are going to be accessed by the end users as it is. So one golden rule is that the time table should be the first presentation table in the subject area. Fact tables can be the last tables in the list because mostly business users start to construct the reports by selecting the first column from the time table. Then they'll select the other attribute columns from the dimensions. At the end, they will select the measures.

There's more...

We can create multiple subject areas in the Presentation layer according to the business requirements, as in the following example:

- Sales for sales representatives
- Sales for sales managers
- Marketing subject area
- Finance subject area

The critical rule in this layer is that subject areas cannot include logical tables from different business models. Every business model should be referenced from a subject area, otherwise the consistency check tool is going to generate an error and change the state of the business model to `Disabled`.

Validating the repository using Consistency Check Manager

Before taking the new repository online, it's required to check the validity of the repository. There's a tool named **Consistency Check Manager** that is used to validate this repository.

How to do it...

1. Whenever you want to save your repository, it's going to ask you if you want to check global consistency or not.
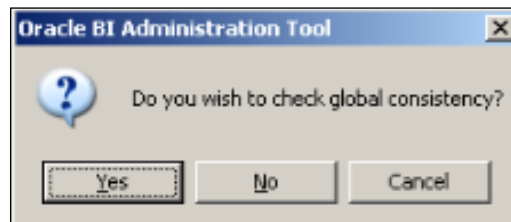
2. We can also trigger this from the **File** menu.
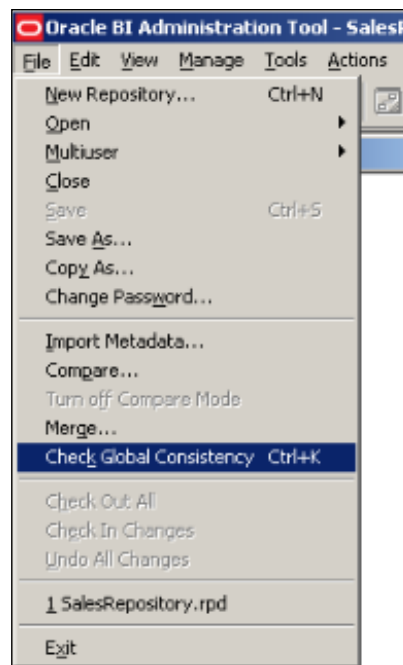
3. The **Consistency Check Manager** tool is going to check the validity of the repository. For example, if the business model is still marked as `Disabled`, it may tell you that the business model is consistent.
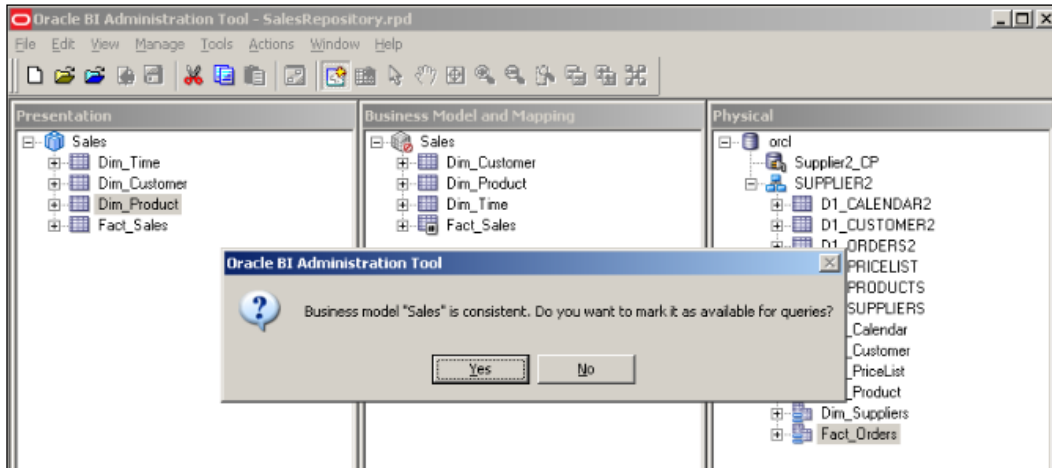
4. We have to be sure about the validity, otherwise when we try to upload it to the server, it's going to generate errors and we won't be able to start the BI server service. If you see a message as shown in the following screenshot, it means the repository is consistent.
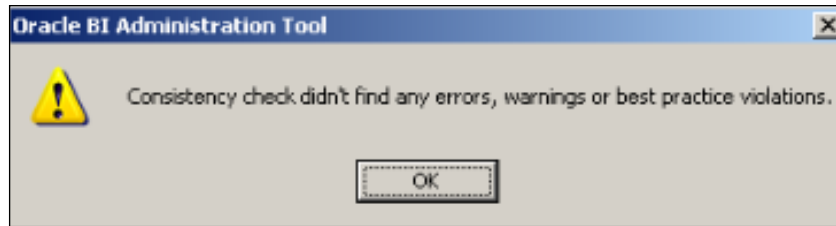
5. We can easily simulate an error by deleting the entire subject area from the Presentation layer, starting the **Consistency Check Manager** tool and seeing the result. It'll give an error about the missing subject area. In this case, the rule that applies is that every business model should be mapped to at least one subject area. Otherwise that business model is not consistent.
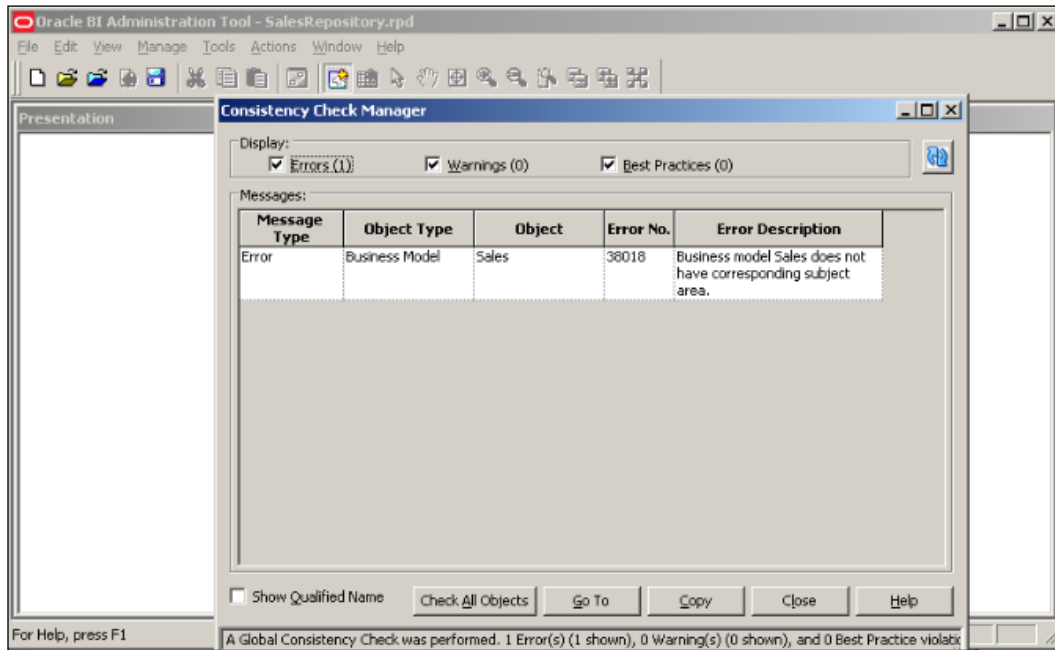
How it works...

The last step before loading the repository to the BI server is to check the server for consistency. Checking the consistency is a step that cannot be skipped. It's always recommended. It's going to check for missing or broken references. So, in every change, we'll have to start the **Consistency Check Manager** tool and see whether there's any inconsistency or not. If you don't check it and try to load the new repository to the BI server and if it's in an inconsistent state, the BI server is going to generate an error and will not start up.

Uploading the repository

After the repository is validated, we upload the repository file to the BI server and start testing to see whether everything works fine or not. We're going to upload the repository by using Enterprise Manager. But before that we'll have to increase the logging level of the user account in order to check the logs that are generated by the end users.

Getting ready

We'll open the Identity Manager tool that was formerly known as Security Manager. The users should be downloaded from the WebLogic Server. But downloading user-identity information can only be done in the online mode. So first we'll upload the repository.

How to do it...

1. Open the **Enterprise Manager 11g Fusion Middleware Control** tool from `http://localhost:7001/em` to upload the new repository.
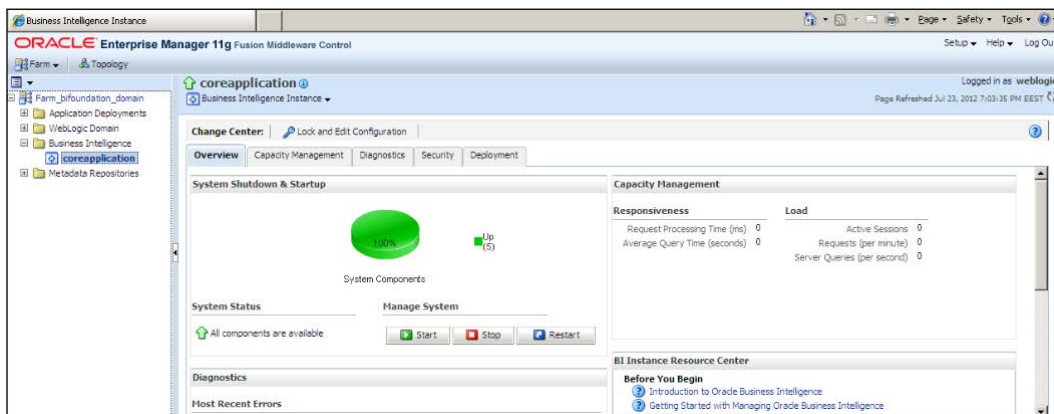
2. When you log in, you'll see the **Deployment** tab and the **Repository** link. This is the page that we're going to upload the repository to. You'll have to show the path of the existing repository and specify the password in **Repository Password**.
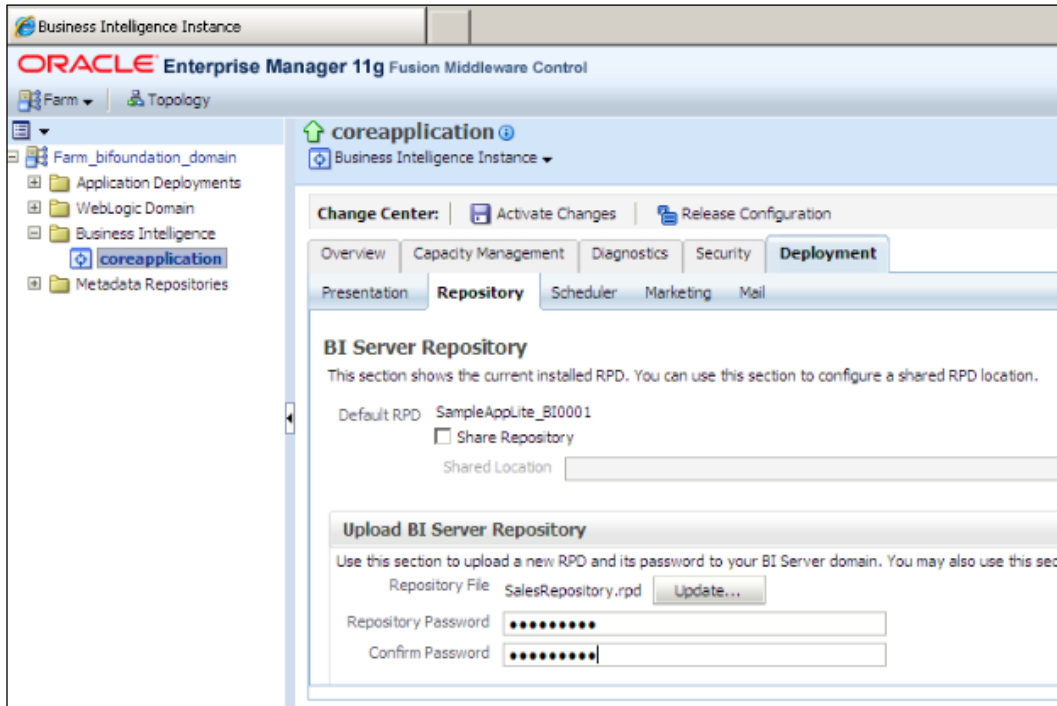
3. Restart the service in order to use the new repository. Now we can run tests, but it's better to increase the logging level and also see the queries that are executed on the physical data sources. When you open **BI Administration Tool** in online mode, you'll be able to download the user information from the WebLogic Server and change the logging level.
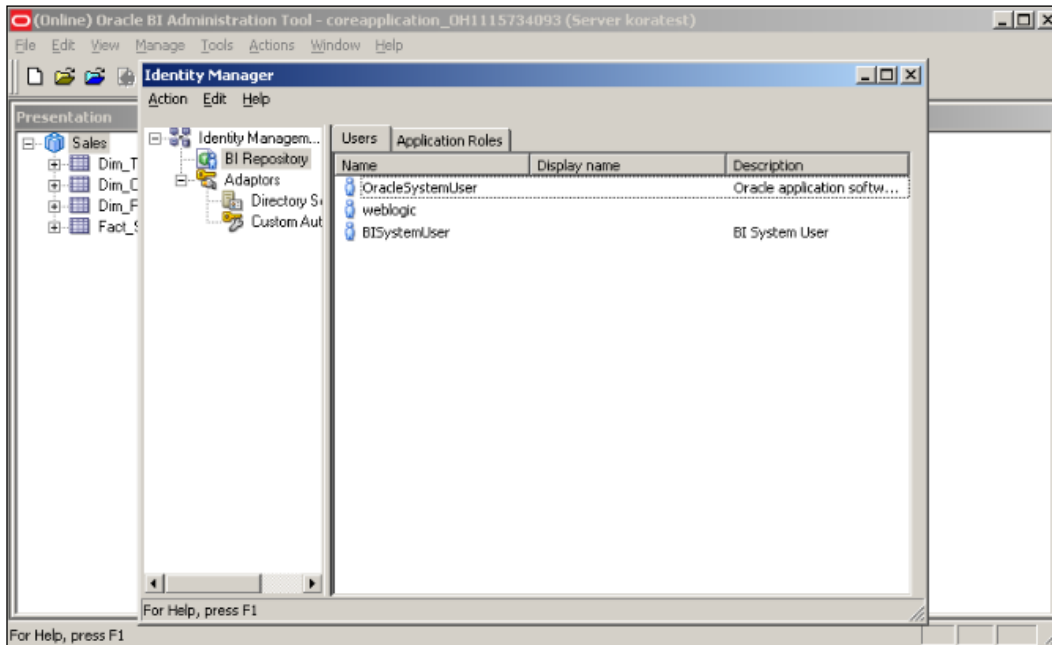
4. Open the properties of the user account that you'll run tests on. In our scenario, we're going to use `weblogic` user account and change the value in **Logging level** to `2`. This level is going to force the BI server to generate both the logical as well as the physical SQL statements.
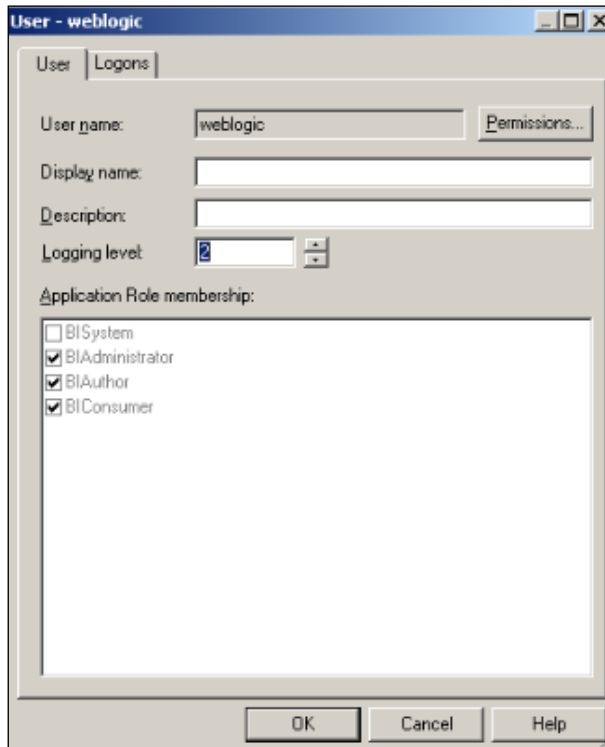
5. Now we can run our tests easily. We're going to use the web browser to access Presentation Service at `http://localhost:7001/analytics`.

6. After you log in and try to create an analysis, you'll see the subject area from the Presentation layer of the repository. Click on the **Subject Area** name.
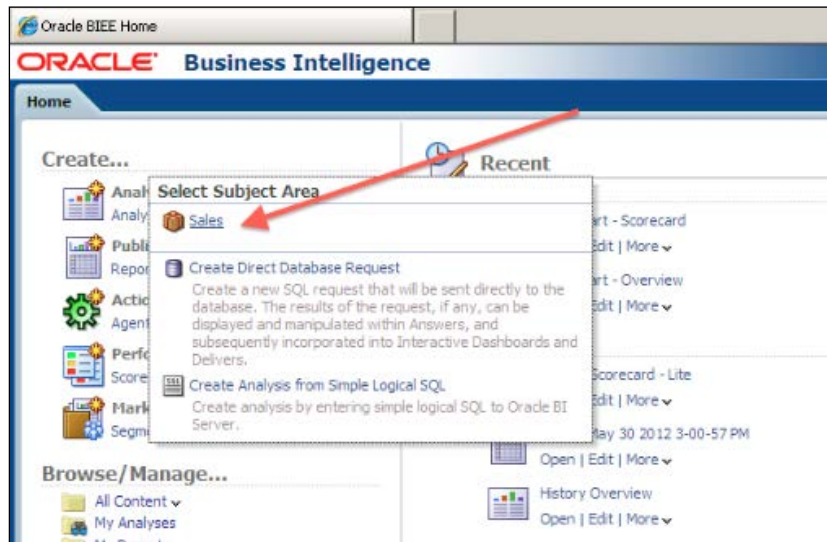
7. It's going to open the Analysis Editor. You'll see that the order of the tables and columns are exactly the same as the order of the subject areas in the Presentation layer. You can easily click on the columns from the **Subject Areas** pane to add them to the **Selected Columns** section.
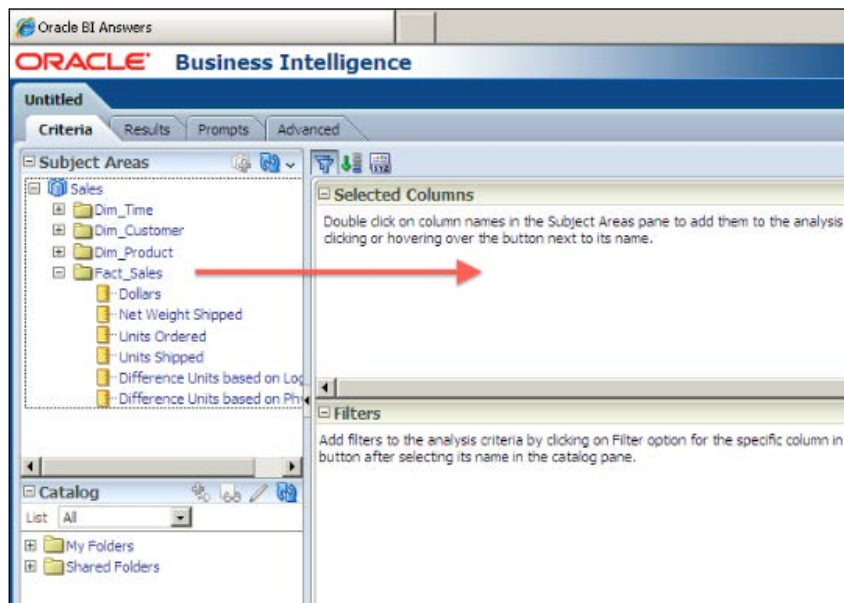
8. Click on the **Results** tab on this page to see the results:
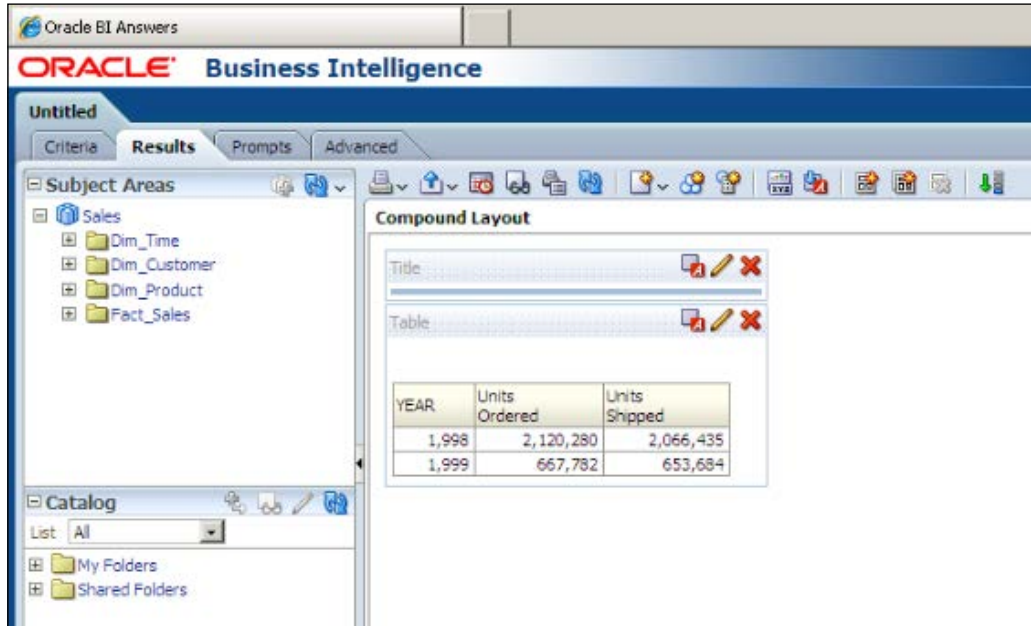
How it works...

After uploading the new repository to the BI server, we'll have to make many tests in order to make sure that everything is fine. Also, logs should be checked.

To make a test, we'll have to log in to Presentation Services and create analysis. After clicking the **Results** tab, we should see the result set as shown in the preceding example. The logical request is going to be constructed from the definition of the analysis at the Presentation Services and it'll be retrieved by the BI server component. The BI server is going to convert the logical SQL statement to a physical SQL statement and execute it against the database. After the result set is generated by the database, that result set is going to pass through the BI server and will be listed as a result at the Presentation Services.

There's more...

After changing the log level of the user that we're running the test on, we can find the execution statistics in the log file. The name of the log file is `NQQuery.log`. The content of the log file can be accessed from the **Administration** link in Presentation Services by navigating to **Administration** | **Manage Sessions** | **View Log**.

2

Working with Logical Dimensions

In this chapter, we will cover:

- ▶ Creating level-based hierarchies
- ▶ Creating parent-child hierarchies
- ▶ Creating level-based measures
- ▶ Creating shared measures
- ▶ Creating presentation hierarchies

Introduction

Logical Dimensions consists of hierarchies and they are created in the Business Model and the Mapping layer. They enable end users to drill down to details, and measures that will be based on levels can also be created. Besides these functions, we can also gain the benefit of using aggregate tables in order to improve query performance.

Logical Dimensions are representations of hierarchical organization of logical columns. There are some common examples of dimensions: `Time`, `Customer`, `Product`, `Channels`, `Employees`, and so on. Setting up these levels of the dimensions is dependent on business requirements. Business users should decide about the granularity levels.

The `Time` dimension might have five levels starting from the GRAND TOTAL level to the DAY level, and we are going to use five-level dimension in our scenario. This can be changed based on the business requirements. After creating this dimension, users will drill down from YEAR to QUARTER, then to MONTH, and then to the DAY level, which is the lowest level in the hierarchy. GRAND TOTAL is the highest level for this dimension that represents the all time data.

GRAND TOTAL
YEAR
QUARTER
MONTH
DAY
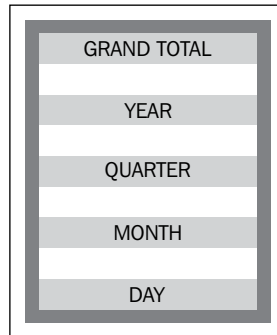
The `Customer` dimension is another common example that contains different levels.

GRAND TOTAL
REGION
COUNTRY
STATE
CITY
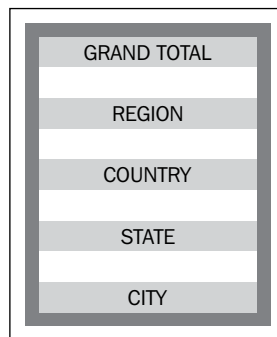
And our last example is about the `Product` dimension.

```
┌─────────────────────────────┐
│  ┌───────────────────────┐  │
│  │     GRAND TOTAL       │  │
│  ├───────────────────────┤  │
│  │   PRODUCT CATEGORY    │  │
│  ├───────────────────────┤  │
│  │     SUB CATEGORY      │  │
│  ├───────────────────────┤  │
│  │       PRODUCT         │  │
│  └───────────────────────┘  │
└─────────────────────────────┘
```
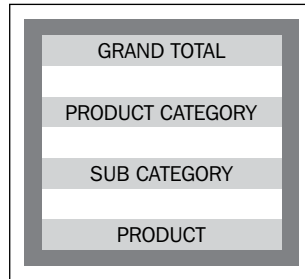
There are two logical-dimension types:

 ▸ **Logical dimensions with level-based hierarchies**: Members of these hierarchies have the same type at a single level. For example, Time, Product, and Customers.

 ▸ **Logical dimensions with parent-child hierarchies**: Members of these hierarchies have the same type at all levels. For example, the Employee and the Manager hierarchy. The members have all the same types at all levels.

Oracle BI Server also supports ragged, skipped level, and time hierarchies that are attributes of the level-based hierarchies.

 ▸ **Ragged hierarchies**: These are unbalanced hierarchies. They do not have the same depth.

 ▸ **Skipped level**: These are the ones that do not have a value for a particular ancestor level. The `Customer` dimension can be an example of this. There's not a State value for some countries so **State level** will be skipped.

 ▸ **Time hierarchy**: In order to gain the benefit of the time-series functions, one `Time` dimension should be specified with a chronological key.

Creating level-based hierarchies

Before starting this task, logical dimension and fact tables should be created in the BMM layer. In addition to this, logical joins should exist including logical keys. If these are not ready, then you should create the logical tables, set the logical table sources, and complete all other tasks that are specified in *Chapter 1, Exploring and Building the Repository*.

How to do it...

1. First, we're going to create the logical dimension with a level-based hierarchy. Right-click on Business Model and select the **New Object** and **Logical Dimension** options. Then select the **Dimension with Level-Based Hierarchy** option.
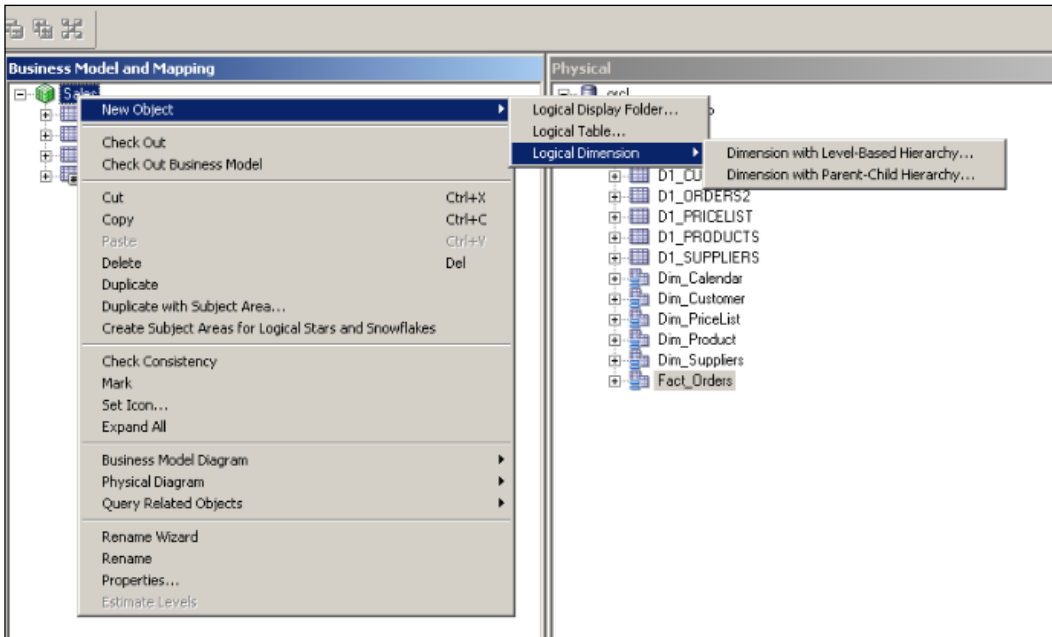
2. The **Logical Dimension – Customer** window will come up and we're going to set the name of the first dimension. We won't be able to select any root level because no level exists at the moment.
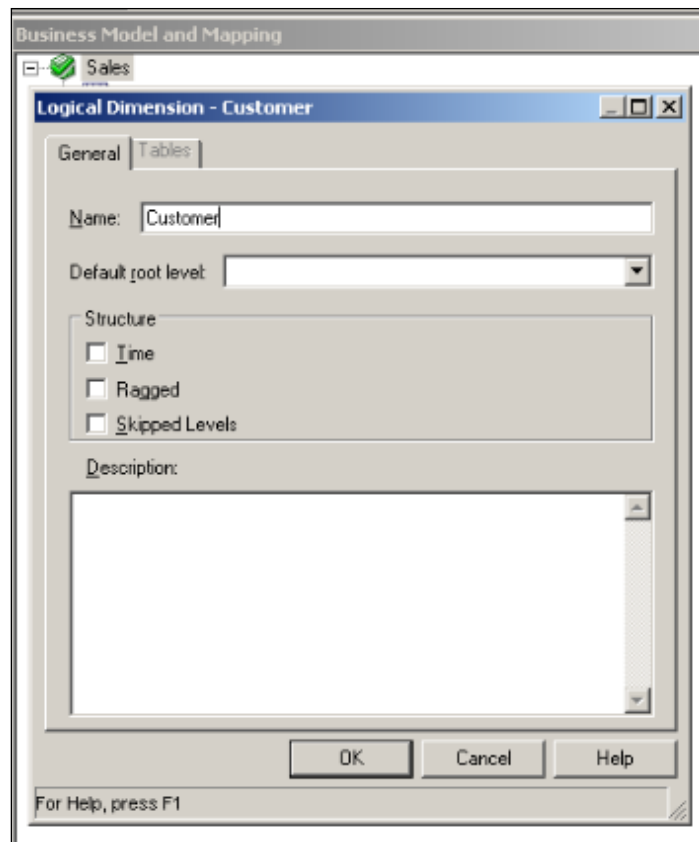
3. After creating the `Customer` dimension, it's time to create the levels one by one. When we check the `customer` table, we'll find out that the following columns exist:

   `Region`, `State`, `City`, and `Customer`

There will be five levels including the GRAND TOTAL level. We're going to click on the **New Object** option on the dimension and select the **Logical Level...** option.
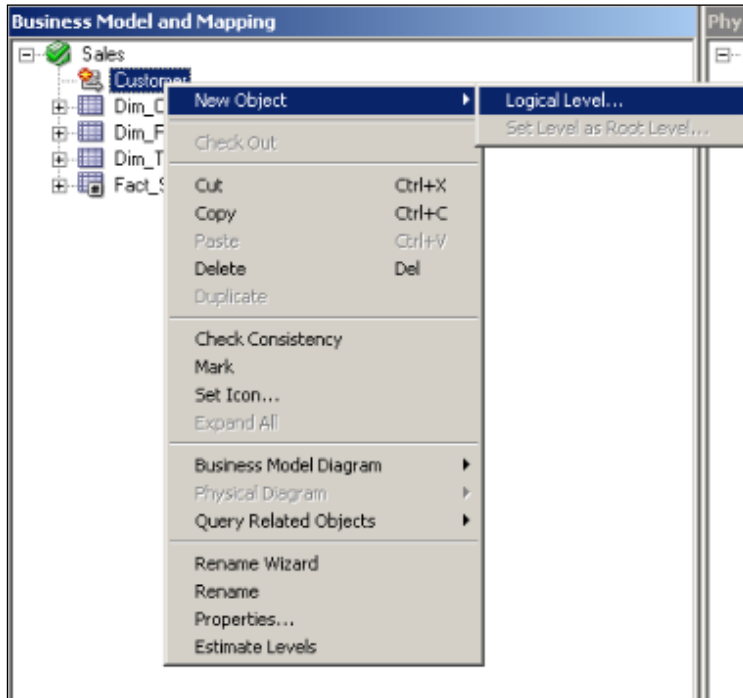
4. The first level will be the root level, which is also called the `Grand Total` level of the `Customer` dimension. Set the name of this level to `Customer Total`. We're going to select the **Grand total level** checkbox. At this moment the **Number of elements at this level** textbox will be disabled and its value will be set to `1` automatically.
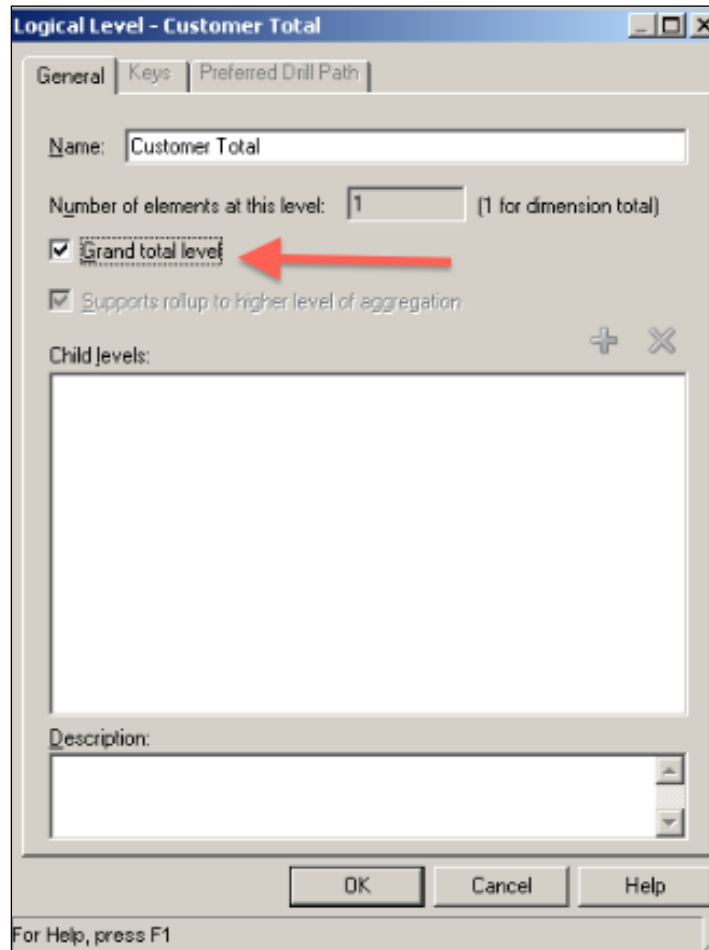
5. The next level after the `Customer Total` level is going to be the `Region` level. This level will be the child level of the root level. That's the reason we're going to select the **New Object** option and the **Child Level...** option on the `Customer Total` level.
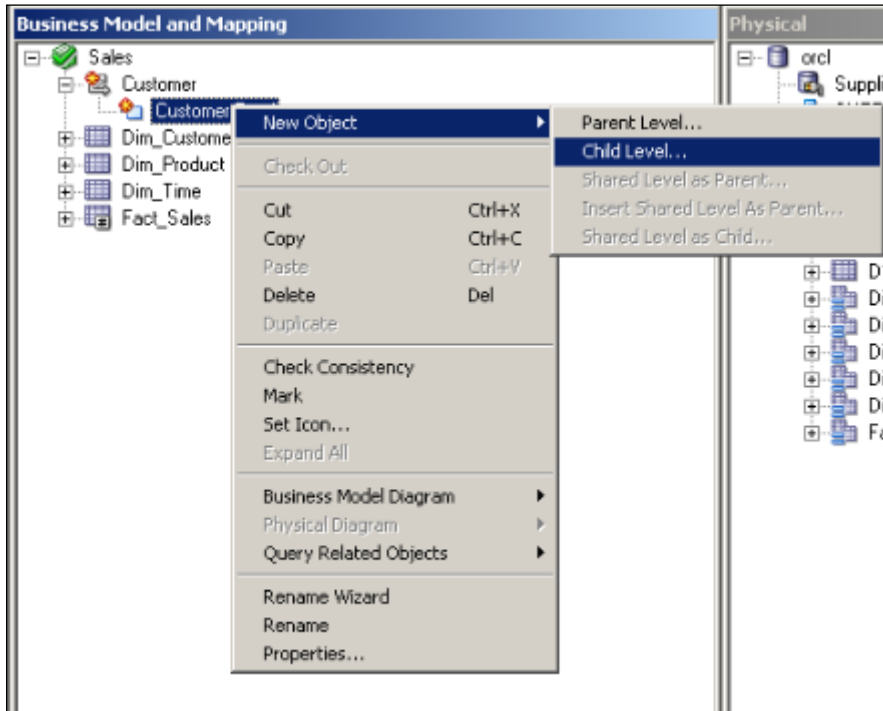
6. When the **Logical Level** window appears on the screen, we're going to set the name of the level and set the number of elements. The number of elements at each level will be used if the aggregate tables are used. We're going to cover the aggregate tables in *Chapter 3, Using Aggregates and the Time Series Functions*.
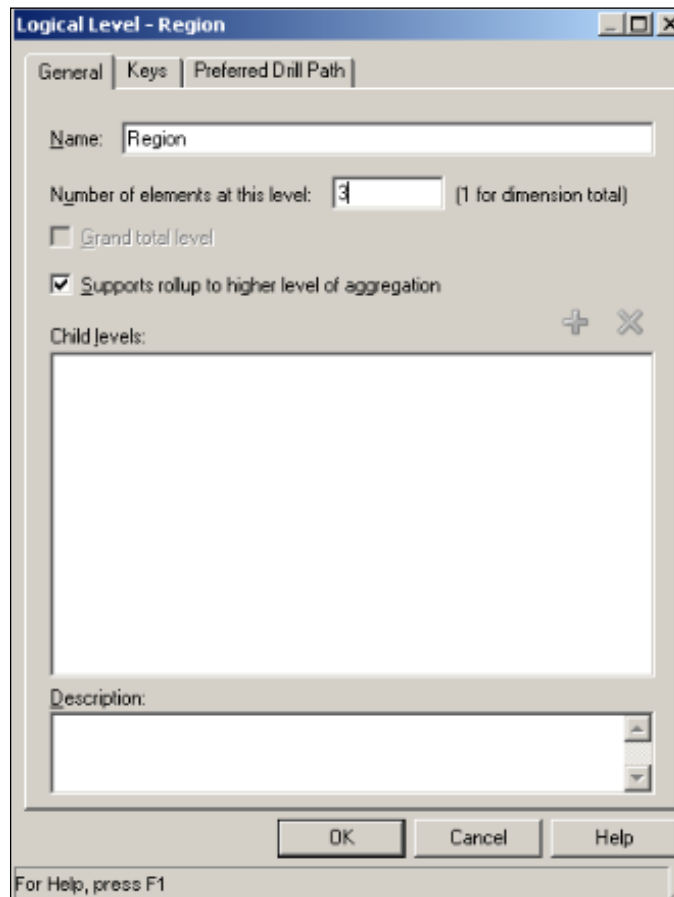
7. Remaining levels will be created the same way we created the `Region` level. We're going to select the `Region` level and right-click on it and then select the **New Object** option. Obviously selecting the **Child Level...** option will bring up the new logical window again.

   As a result, you'll see all the levels in the following screenshot:
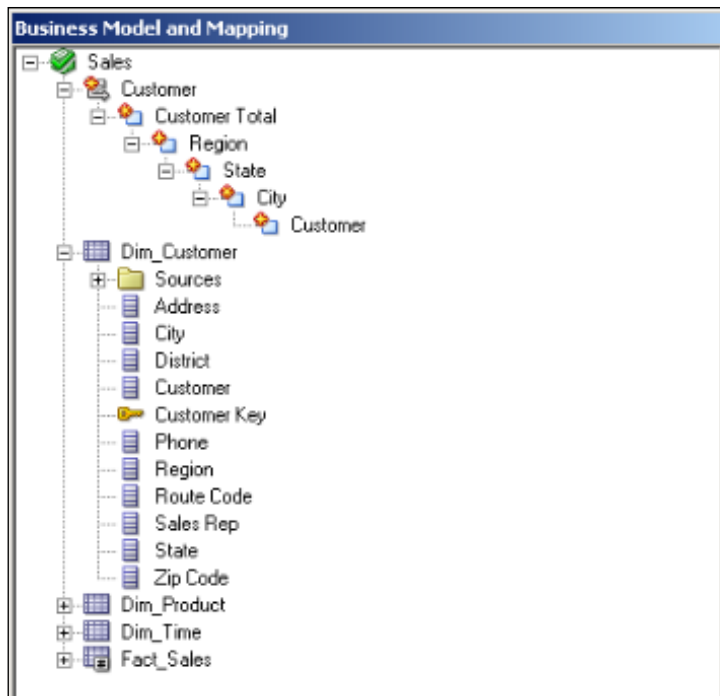
8. The `Customer` dimension is created with five levels including the `Grand Total` level. Now we'll have to define the Level and Logical Column mapping. This can be achieved in two ways. You're going to see the drag-and-drop method in the following screenshot. Drag the `Region` logical column on to the `Region` level.

Business Model and Mapping

- Sales
  - Customer
    - Customer Total
      - Region
        - State
          - City
            - Customer
  - Dim_Customer
    - Sources
    - Address
    - City
    - District
    - Customer
    - Customer Key
    - Phone
    - Region
    - Route Code
    - Sales Rep
    - State
    - Zip Code
  - Dim_Product
  - Dim_Time
  - Fact_Sales
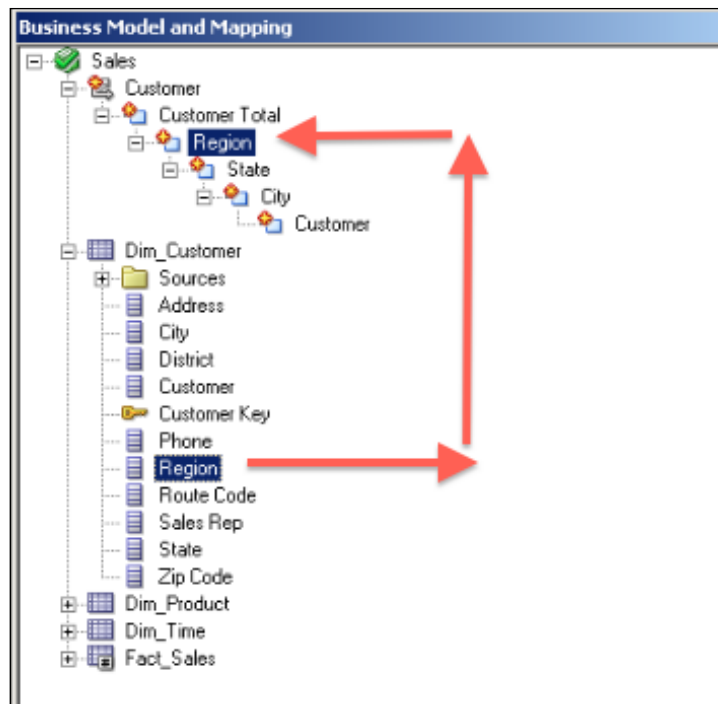
9. The second method of the Level and Logical Column mapping is the manual method. You'll have to open the properties of the logical column. In our scenario it will be the `State` logical column. Then you're going to select the `State` level from the **Logical Level** drop-down list.

Logical Column – State

General | Column Source | Aggregation | Levels

Logical Dimension	Logical Level
Customer	State
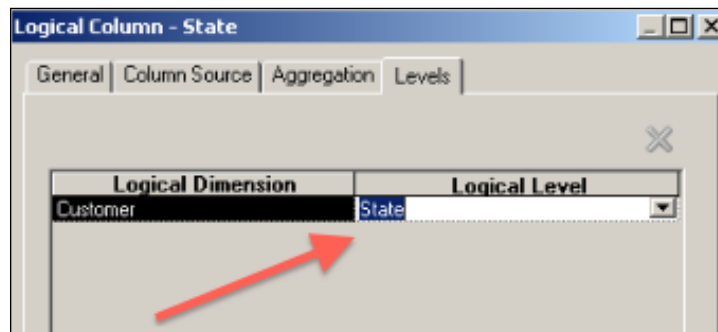
10. After mapping the logical columns with the levels, there will be one more step. The keys and their mapping columns should be defined at each level one by one, except the GRAND TOTAL level, to ensure uniqueness. So you'll see the **Keys** tab when you double-click on the level. You'll see the **Logical Level - Region** properties in the following screenshot. We'll create a key definition by specifying the **Key Name** option and selecting the proper column from the **Columns** drop-down list. There's also one important setting that is called **Use for Display**. This checkbox should be selected if you want to drill down through this level. If there's no key selected for display, then the users won't be able to drill down.
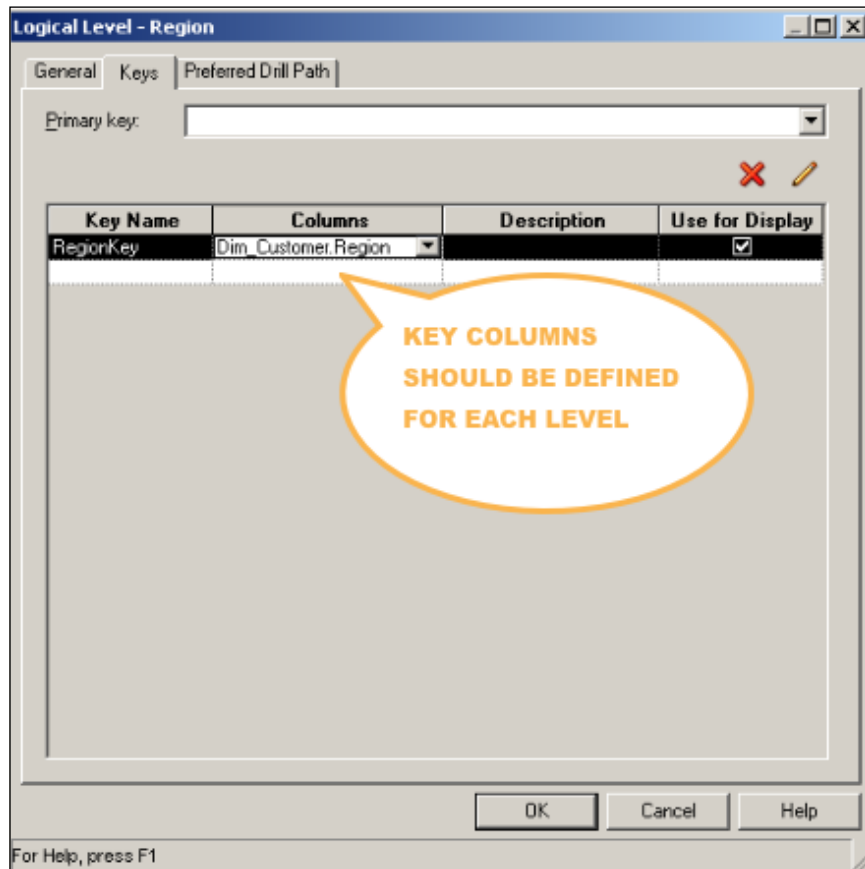
At the end, the `Customer` dimension will look as in the following screenshot. All the levels and the logical mappings are defined and all the keys are created at each level.
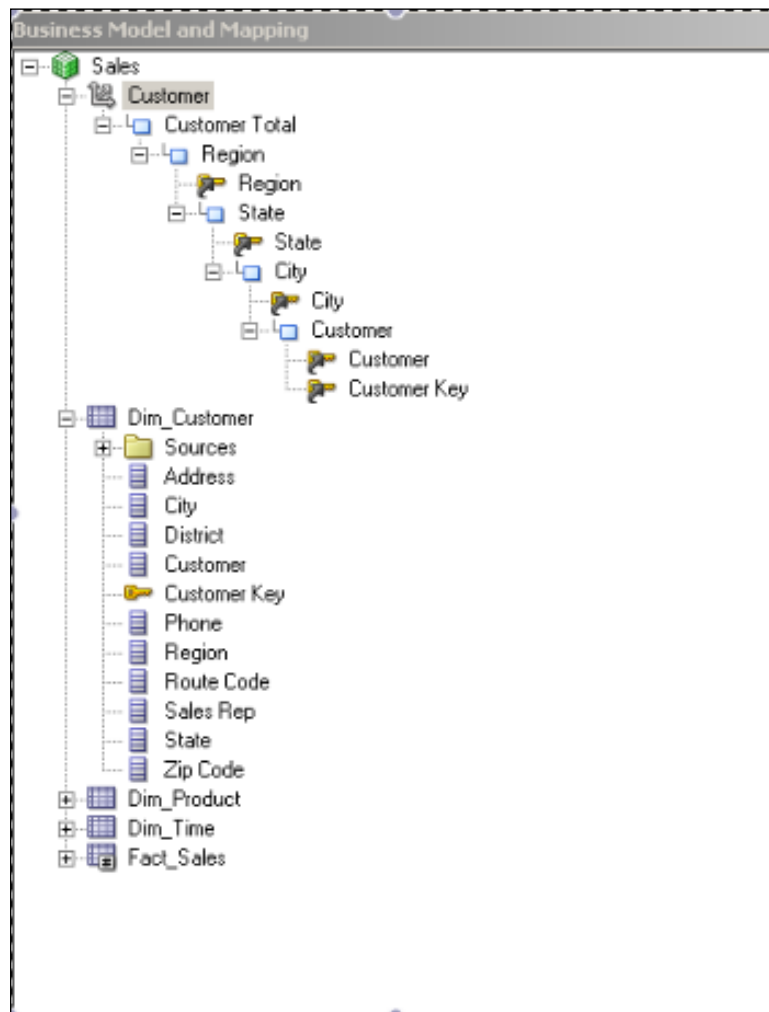
11. Now we're going to create the `Time` dimension and all of the steps are going to be the same. The only difference is that we have to select the **Time** checkbox in the **General** tab of the `Time` dimension in order to gain benefit of the **time-series functions.**
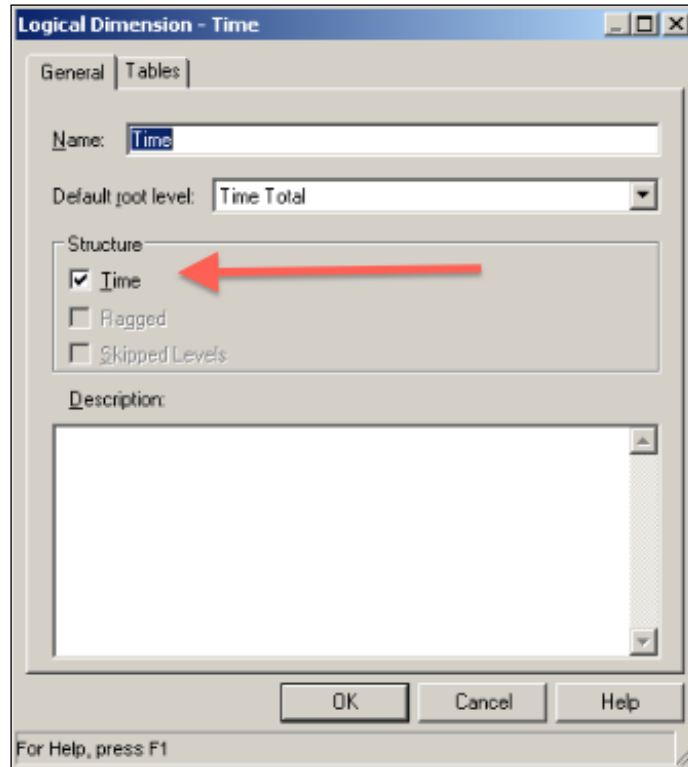
12. Once it's selected, you'll double-click on the **Logical Level** option and notice that the **Keys** tab content is changed. The new column name is called **Chronological Key**. This key is again needed by the time-series functions. We're going to select this checkbox to make the dimension ready for the time-series functions. These functions will be covered in *Chapter 3, Using Aggregates and the Time Series Functions*.
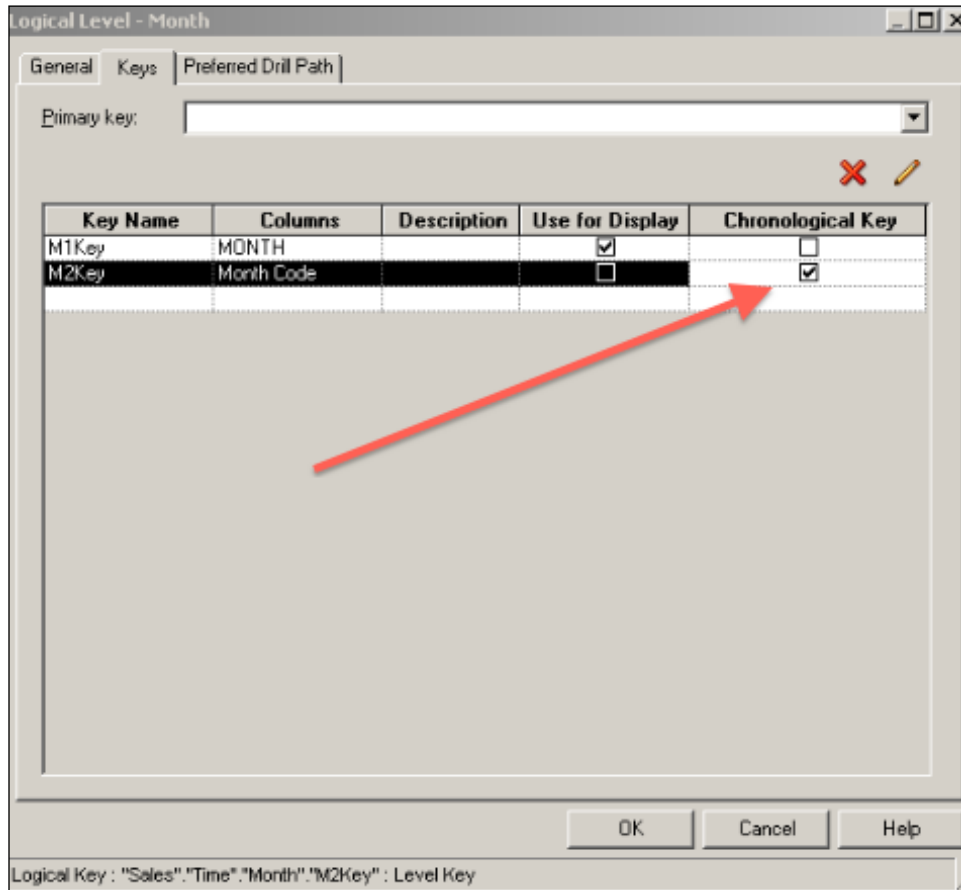
The `Time` dimension will be as it is in the following screenshot. It'll have four levels:

`Grand Total | Year | Month | Day`
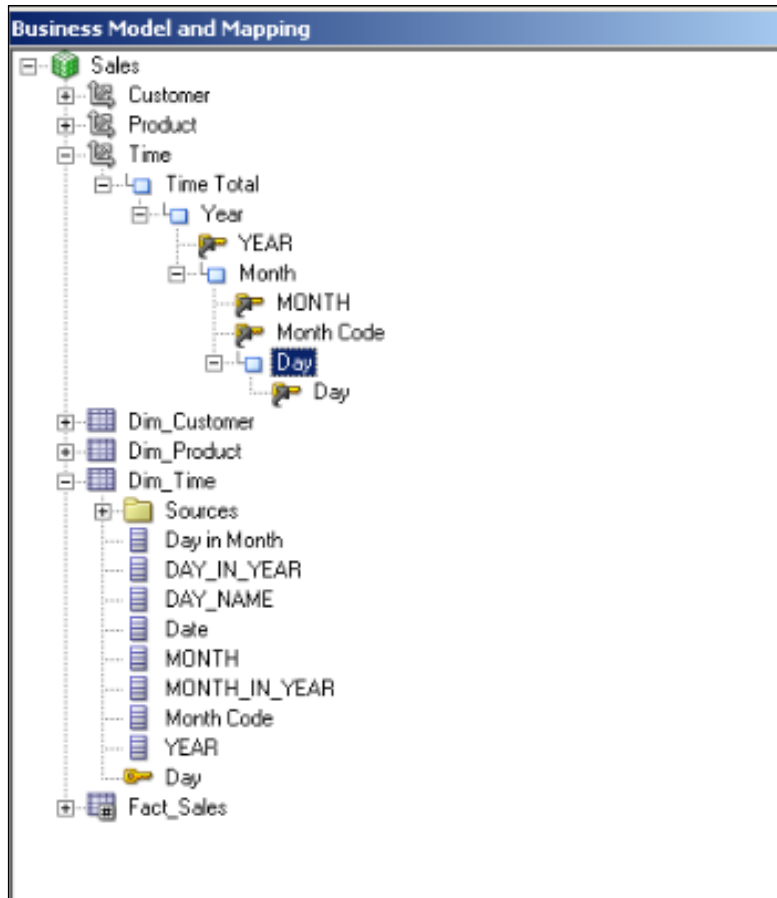
The `Product` dimension hierarchy has five levels and it is specified as follows:

`Grand Total | Type | Sub Type | Generic | Specific`
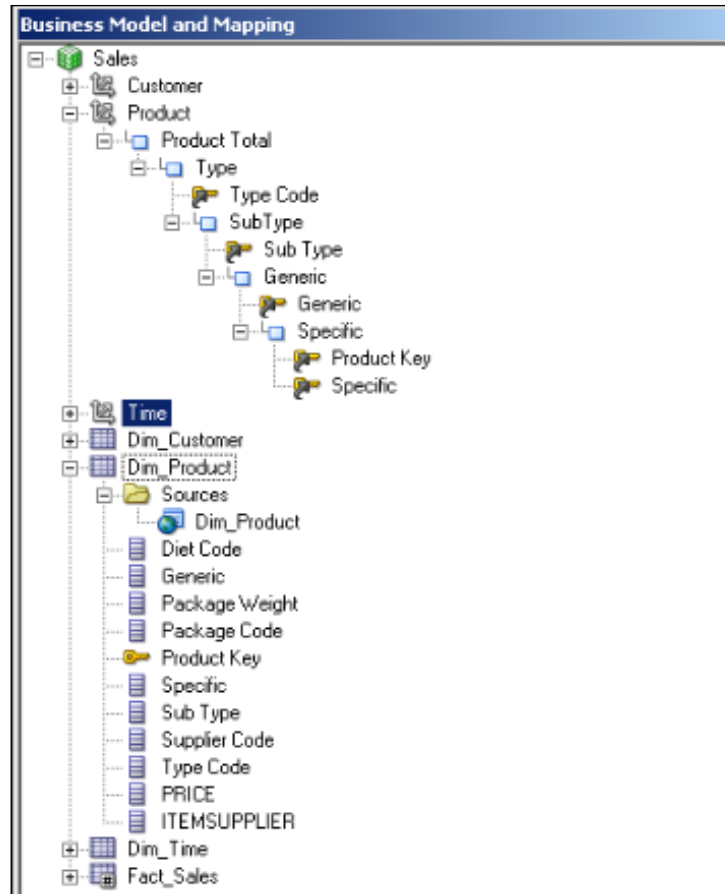
How it works...

Logical dimensions with level-based hierarchies are going to enable end users to drill down to the analysis in Presentation Services. When you create any analysis and use a logical column that is mapped to a logical level, you'll see that the value is going to be displayed with a hyperlink. So when the user clicks on the hyperlink, it's going to drill down to a lower level. The default action of the columns is set as **Drill Down** if a dimension object exists. Of course this default behavior can be changed from the Analysis Editor in Presentation Services.

There's more...

Another benefit of dimension objects is the use of level-based measures. This will bring a new feature to the analysis in order to create comparison reports. You'll easily create new calculation measures. Additionally, if you want to improve the query performance by using the Aggregate tables, dimension objects will be needed. These subjects also will be covered in *Chapter 3*, *Using Aggregates and the Time Series Functions*.

Creating parent-child hierarchies

In order to demonstrate the creation of the logical dimension with parent-child hierarchies, we're going to use another data source and another business model. So three new tables are imported into the Physical layer. And all the physical joins and keys are defined. Additionally, a new business model is created and the logical joins are also created in the Business Model diagram. So all the prerequisite steps are done before creating the parent-child hierarchy. You'll notice that there's a logical table called `EmployeeManagers` and there're 2 columns that will be used in the example. These columns are `EMPLOYEE_ID` and `MANAGER_ID`.

The new business model can be found in the following screenshot:
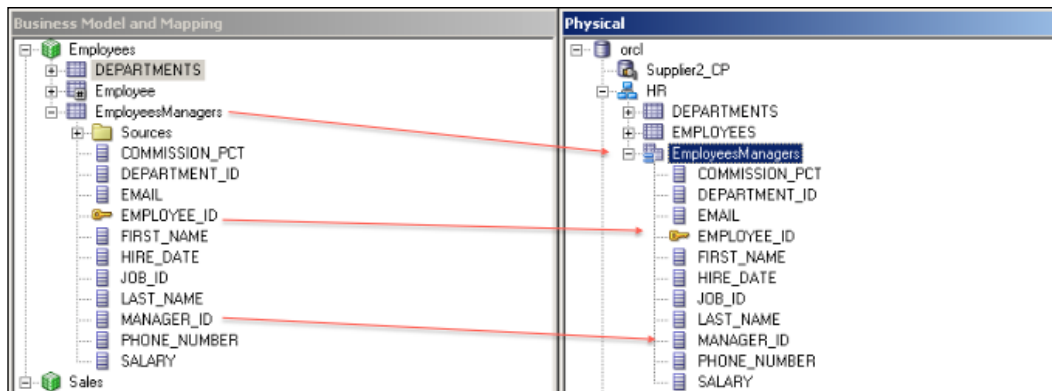
How to do it...

1. Again we're going to right-click on the Business Model and select the **Dimension with Parent-Child Hierarchy...** option.
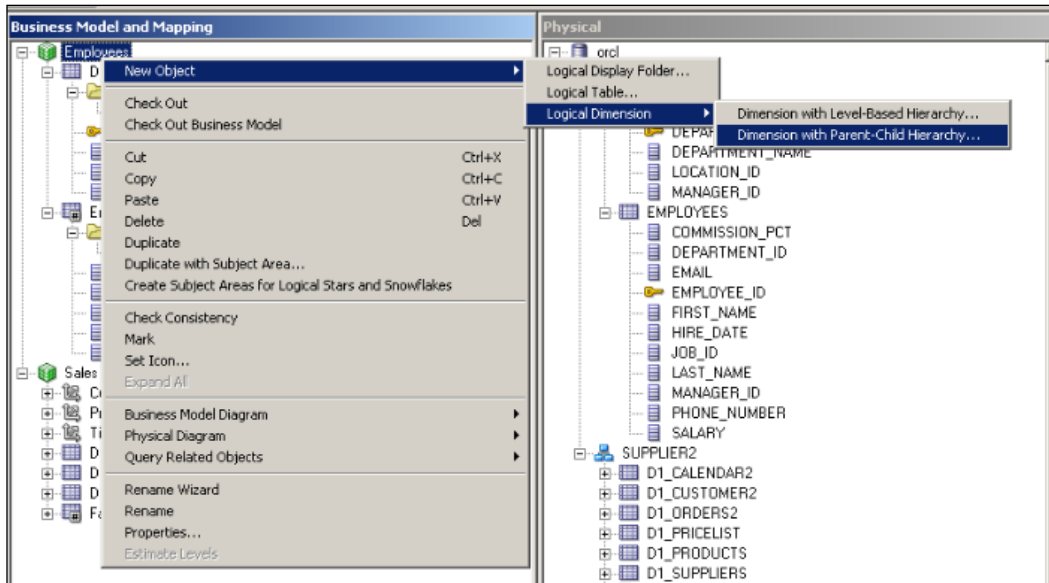
The new **Logical Dimension** window will appear on the screen.

2. First of all, we're going to set the name of the new dimension. Then it's going to ask to select the parent (**Parent Column**) and child key (**Member Key**) columns. In our scenario, we're going to set the `EMPLOYEE_ID` key column as the Member Key and `MANAGER_ID` column as the Parent Column.

3. Now Oracle BI Server will need a parent-child relationship table. This table will have four columns:

   ❑ Member Key

   ❑ Ancestor Key

   ❑ Distance

   ❑ Is Leaf

If you have already created this table manually, then we'll have to select it. Or OBI Server can create it and load the data into it with a wizard. We're going to use the second way. We'll click on the **Parent-Child Settings...** button.
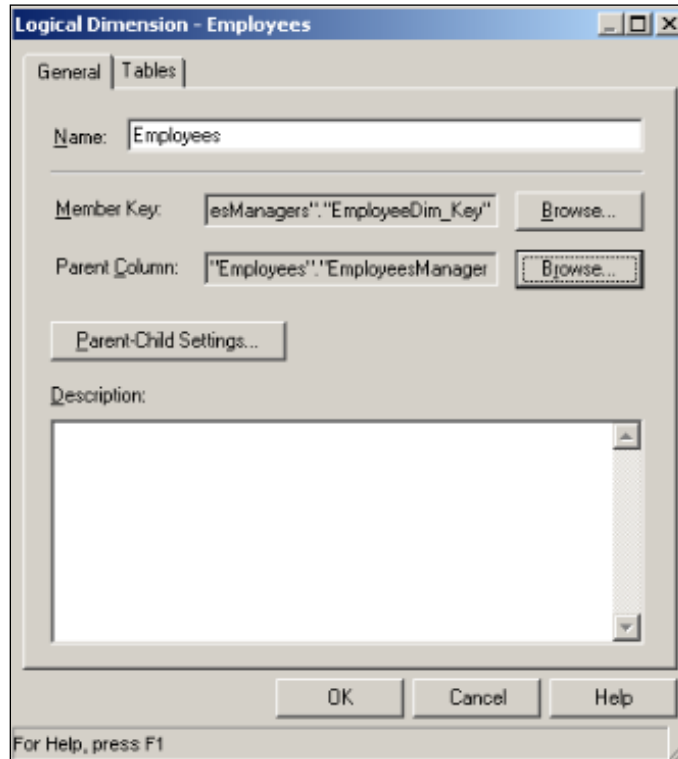
4. The **Parent-Child Relationship Table Settings** window will come up in order to make a selection or just to create this relationship table. You'll see the **Create Parent-Child Relationship Table** button. This button is going to trigger the wizard.
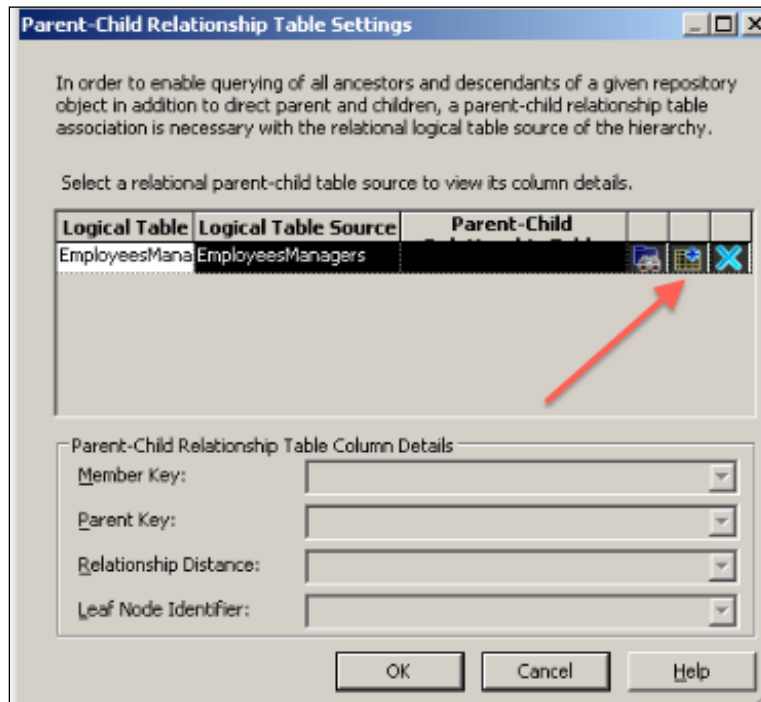
5. This 3-step wizard is going to ask to specify the path for the scripts. You'll notice that there will be two scripts. One of them is the **DDL** script for creating the relationship table. The other is for loading the data from the `EmployeesManager` table.
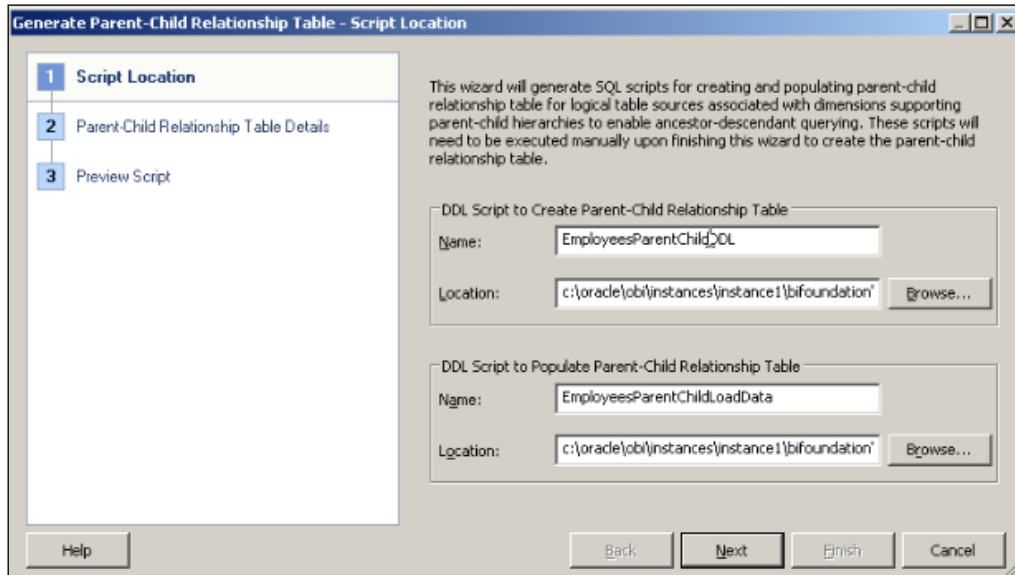
6. You'll select the physical database and the schema where the scripts are going to be executed in the second step of the wizard.
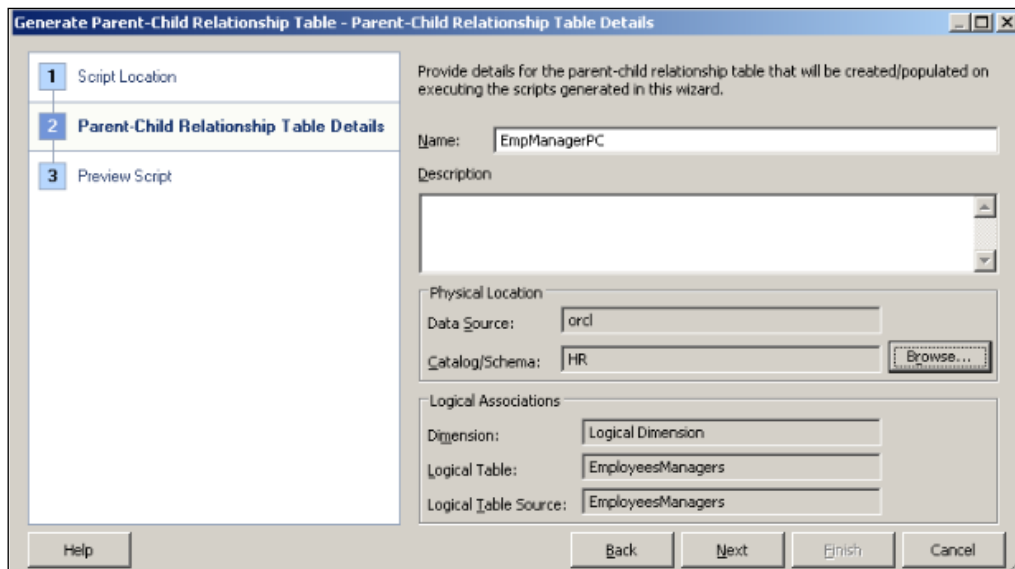
7. The last step of the wizard contains the script content. You can easily preview the scripts.
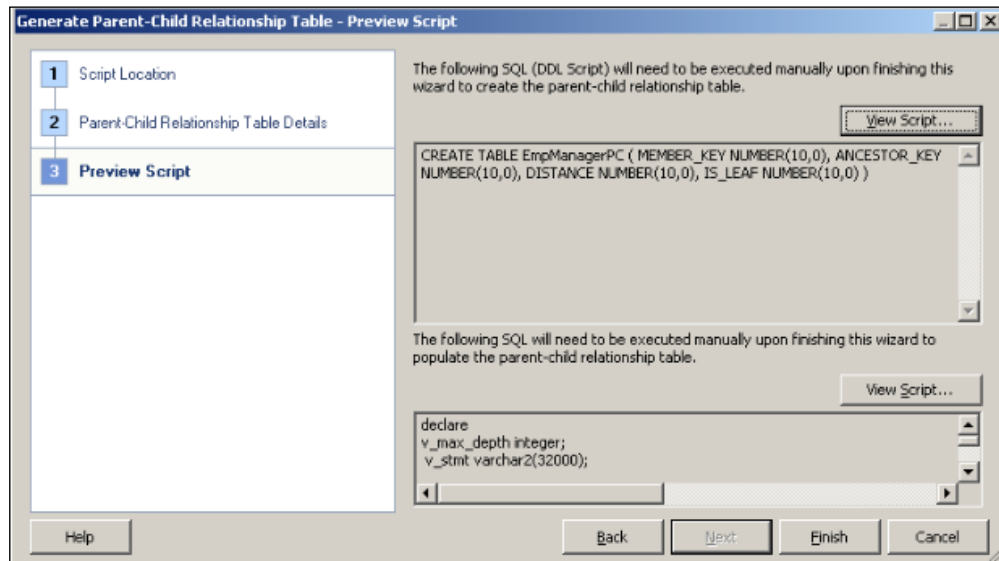
8. Clicking on **Finish** is going to bring the **Parent-Child Relationship Table Settings** window back, and you'll find out that all the column mappings are done automatically.
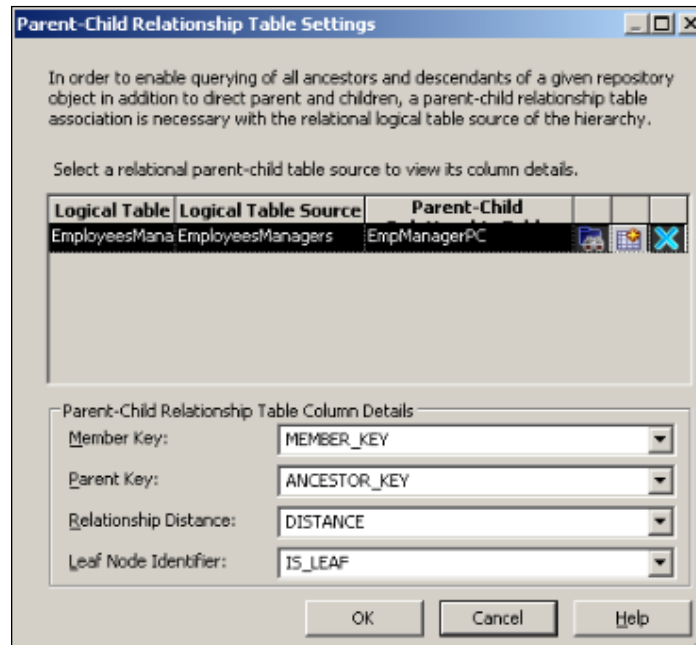
9. When the creation of the `Employee` dimension is completed, you can check if the parent-child relationship table exists in the physical layer or not. Actually, the wizard creates this table and loads the data into it. And then it also imports this table to the physical layer and creates the physical joins as well. You can see the details in the following screenshot:
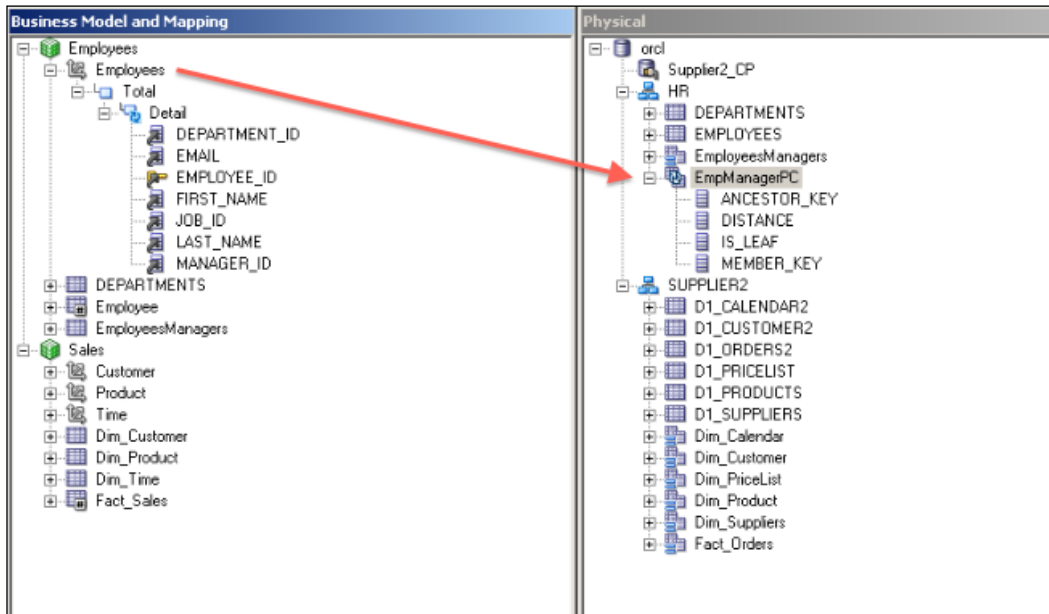
How it works...

Dimension with the parent-child hierarchy is very similar to the dimensions with level-based hierarchies. The only difference is that all the members at all levels are the same type in the parent-child hierarchy. The benefits are also same.

There's more...

Logical Dimension Hierarchies can be added to the Subject Area in the Presentation layer. So the users can see the hierarchies before they drill down. This is a new feature that is introduced in the OBI 11*g* Server.

Creating level-based measures

A **level-based** measure is a column whose values are calculated to a specific level aggregation such as `YearRevenue` or `StateRevenue`. For example, if the business users want to calculate the total revenue for the GRAND TOTAL level of the `Customer` dimension, we're going to create a level-based measure, which will be mapped to GRAND TOTAL level of the `Customer` dimension and this will calculate total revenue across all regions.

In order to create the level-based measure, dimensions and hierarchies should be created before. Then we're going to create a new logical measure column and map it to a specific level.

How to do it...

1. First step will be creation of a logical measure column. So we're going to right-click on the fact table and select the new **Logical Column** option.
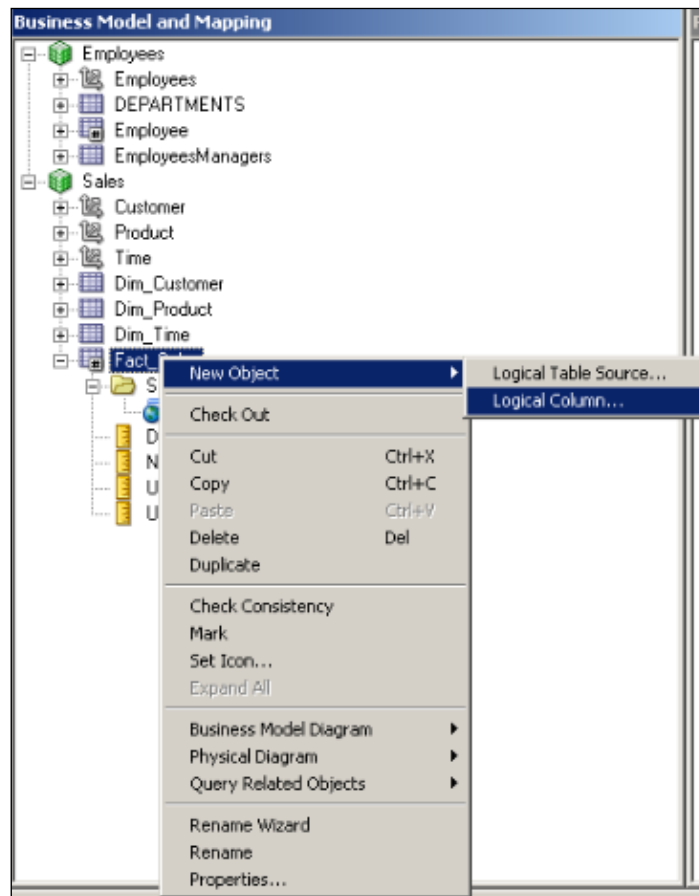
2. When the **Logical Column** window comes up, we're going to write the name of the measure column. `Region Total Dollars` is used in the example.
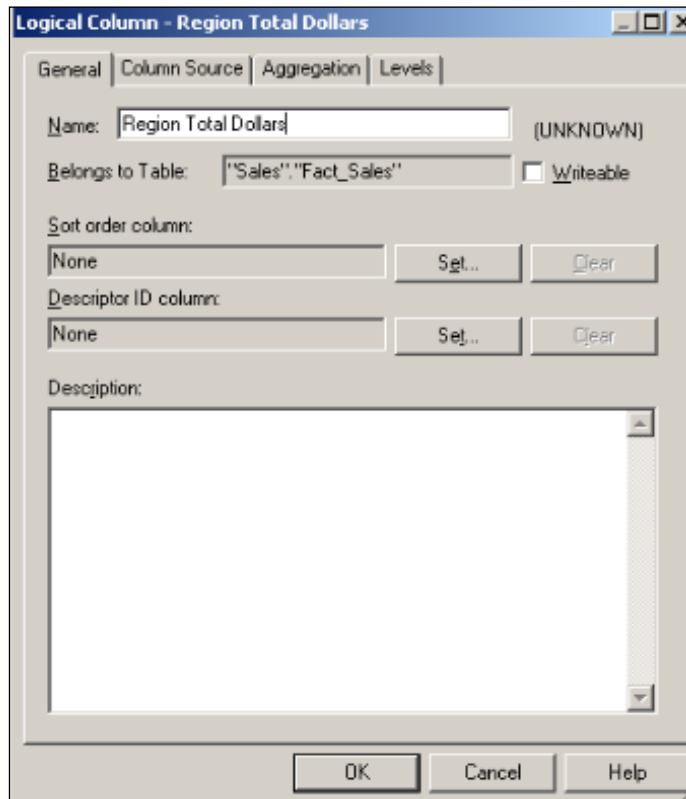
3. Instead of mapping this logical column with a physical column, we're going to just map it to another existing logical column. In our case it'll be the `Dollars` logical column. So we're going to check the **Column Source** tab and select the **Derived from existing columns using an expression** option. After that, logical column `Dollars` is going to be the source for the `Region Total Dollars` **column**. At this moment, this logical column will definitely be the same as `Dollars`.
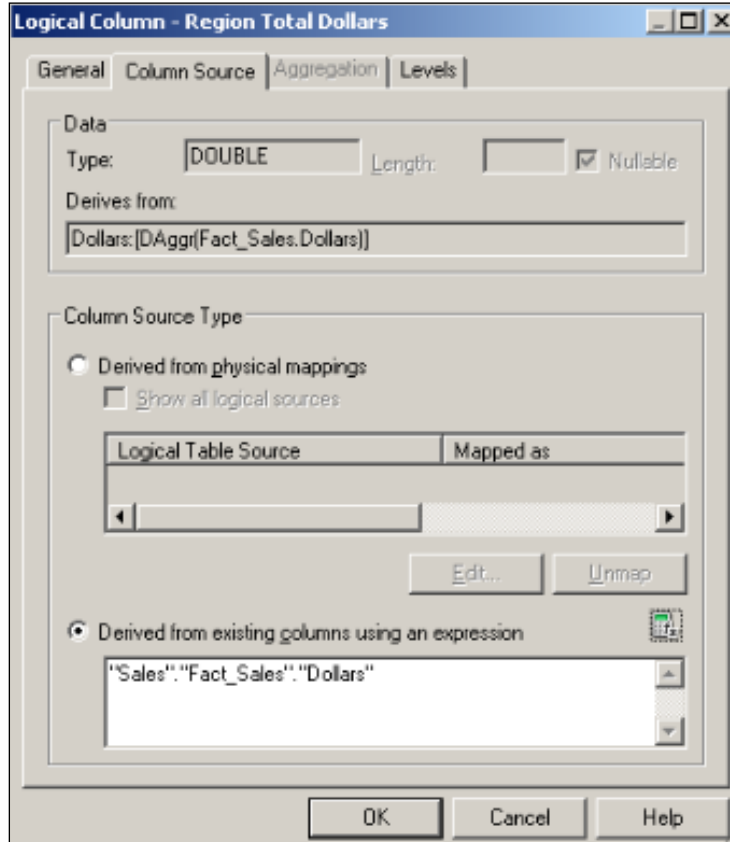
4. But when we clicked on the **Levels** tab, we'll see all the dimensions. We're going to select a level in one dimension. In our case the **Region** level is selected for the `Customer` dimension.
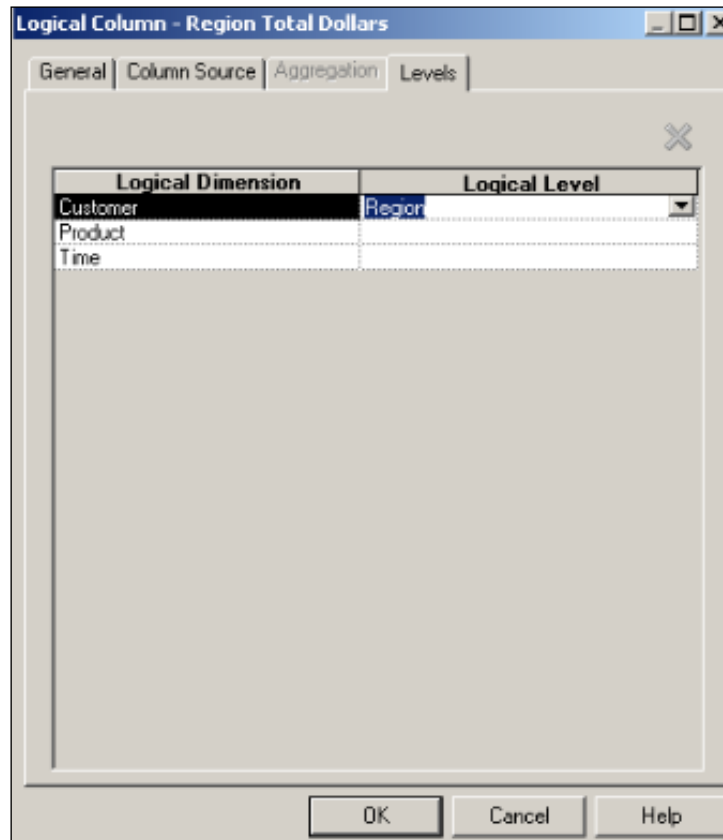
5. Clicking on **OK** will take you to the Business Model layer again. You'll notice that the `Region Total Dollars` logical column is mapped with the logical level, **Region**, in the **Customer** dimension automatically.
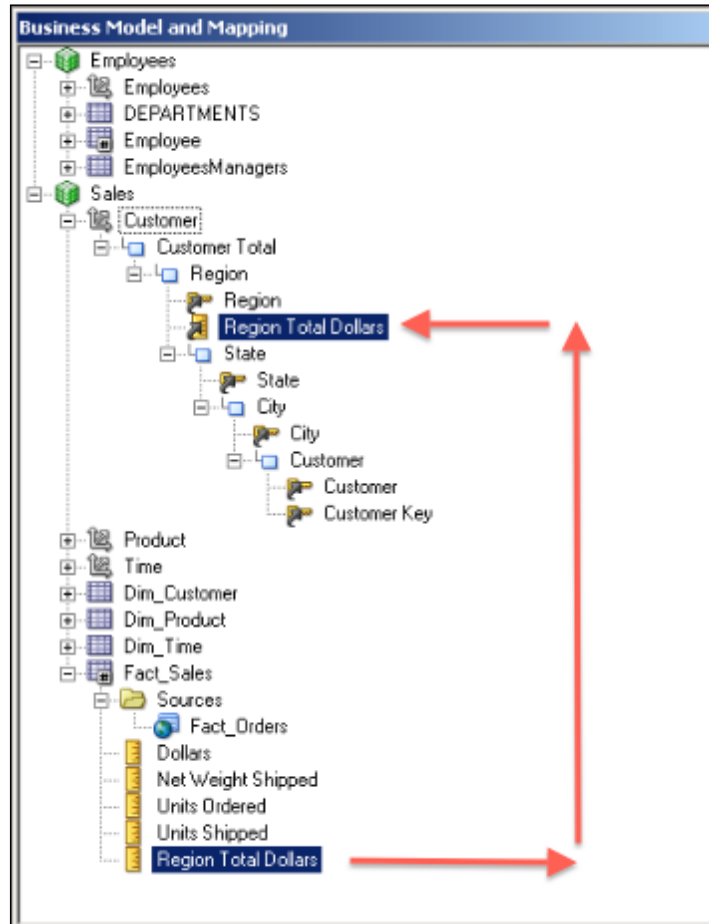
6. Just to have one more example, we can easily create one more logical column which is referencing the `Dollar` logical column. But this time we're going to map this to another level in another dimension. The last example is the `Year Total Dollars` column.
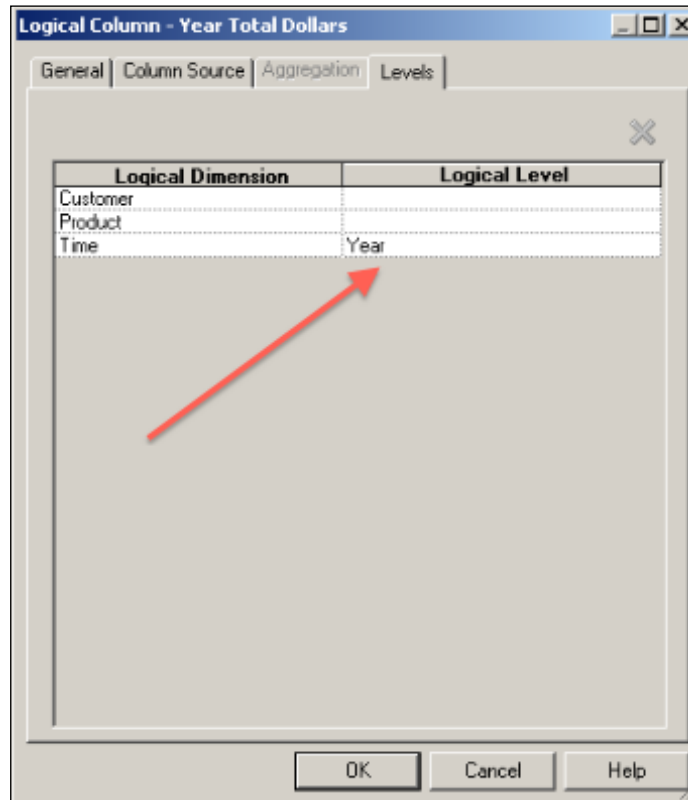
7. And again the measure is automatically mapped to the corresponding level.
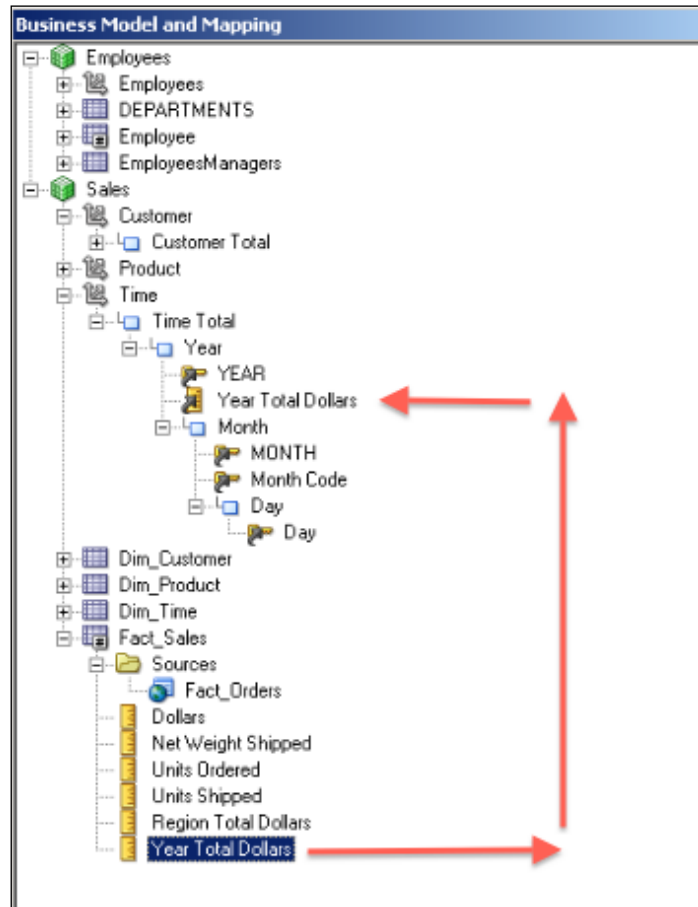
8. We're going to add the new logical columns to the subject area. So end users will access these two new measures from Presentation Services.
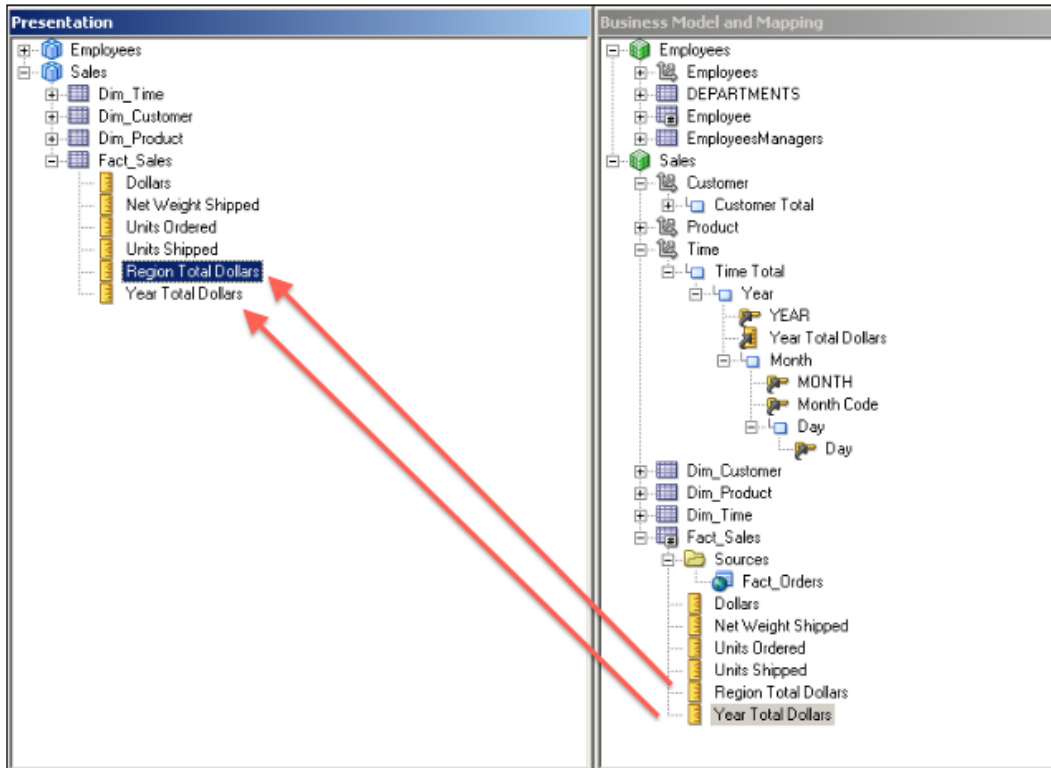
How it works...

After creating new measures, we'll be easily accessing these two measures from the Analysis Editor. Here's an example that contains the new measure. Four columns are already selected:

▸ YEAR

▸ MONTH

▸ Dollars

▸ Year Total Dollars

The new measure shows the total value that is aggregated at the YEAR level. You'll see that the total amount of dollars for January is 3,568,665 and the total value for the year is 47,748,591.

There's more...

When it comes to creating reports in Presentation Services, we'll have to think about how to measure the sales or productivity in an organization. We should not only show the values or amounts. These values should be comparable and measurable.

Creating shared measures

Shared measure is a measure that is calculated by using a level-based measure. There's no difference when you compare it with other logical measures.

How to do it...

1. We're going to create a new logical column in the fact table and again we're going to define a formula. In the formula, we're going to use a level-based measure as well. The formula I used in my demonstration is as follows. It's going to calculate the percentage of a month level sales amount with the total year level sales.

    ```
    (Sales.Fact_Sales.Dollars / Sales.Fact_Sales.Year Total Dollars )
    * 100
    ```

    This time we're not going to change anything in the **Levels** tab. We're going to leave the settings as it is.
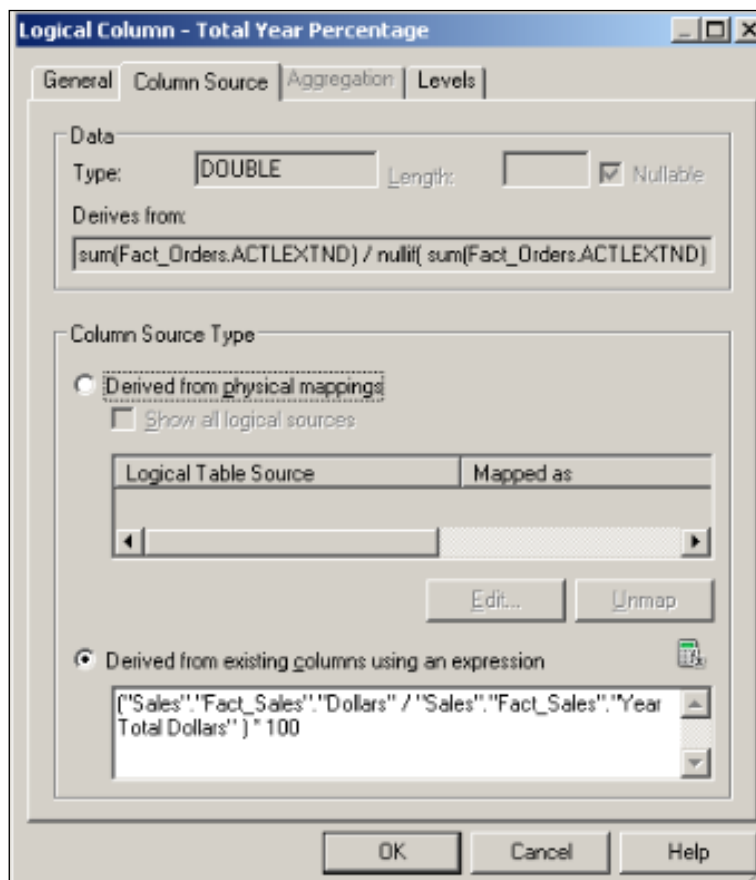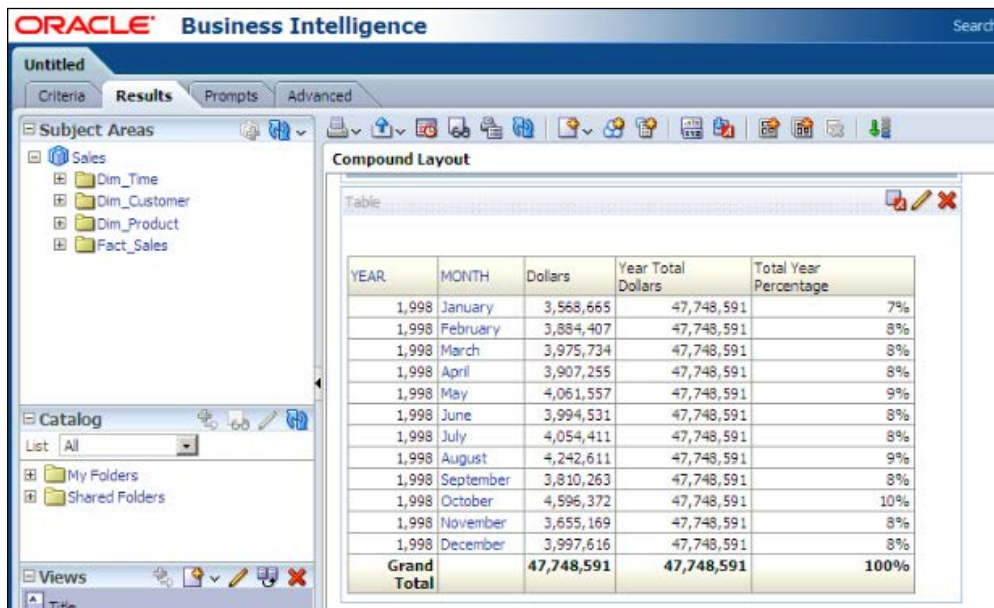
2. As usual, the new logical column should be added to the **Presentation layer**. So we're going to drag it onto the corresponding fact table.
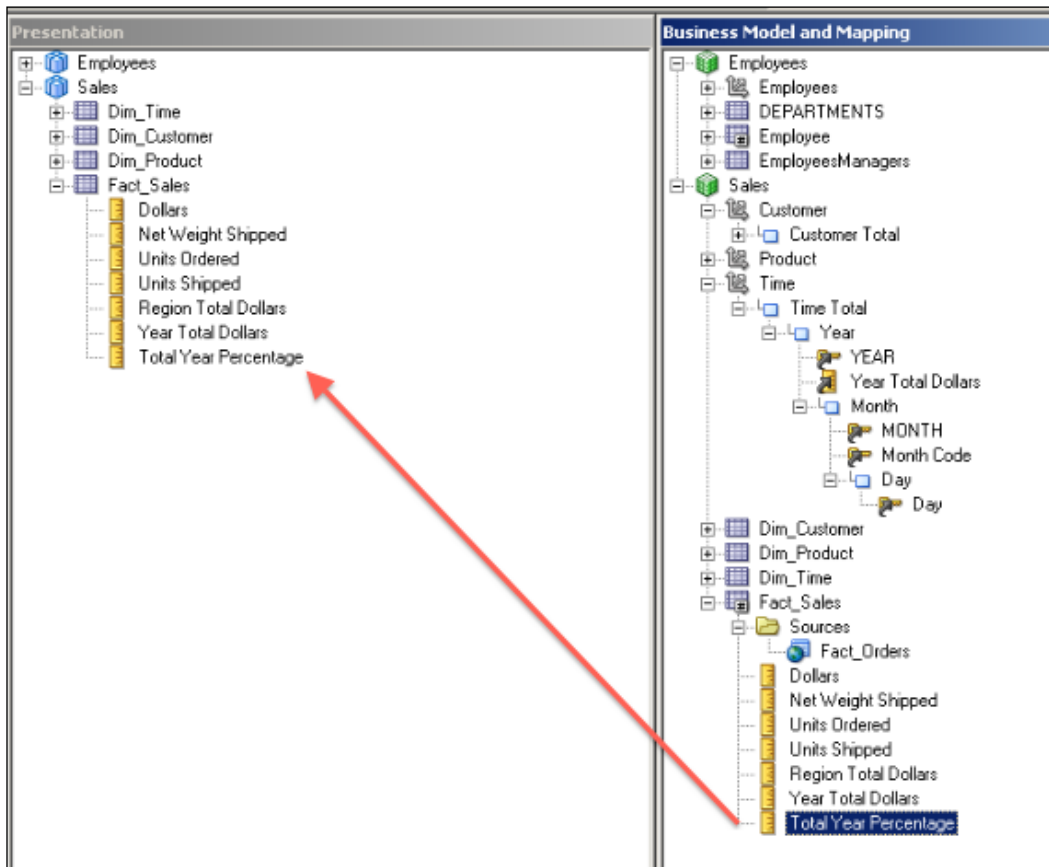
How it works...

After starting to create a new analysis, we're going to see the new measures and use them in the analysis. As you see in the following example, there are five different columns:

- ▸ `YEAR`
- ▸ `MONTH`
- ▸ `Dollars`
- ▸ `Year Total Dollars`
- ▸ `Total Year Percentage`

Creating presentation hierarchies

In the OBIEE 10*g*, this feature was not supported. Exposing the hierarchies enables end users to learn about the levels of the dimensions without drilling down. This feature is introduced with OBI 11*g*.

How to do it...

1. We're going to just drag the `Customer` dimension on to the `Customer Presentation` table. You'll see that the hierarchy is exactly the same as it is on the BMM layer.
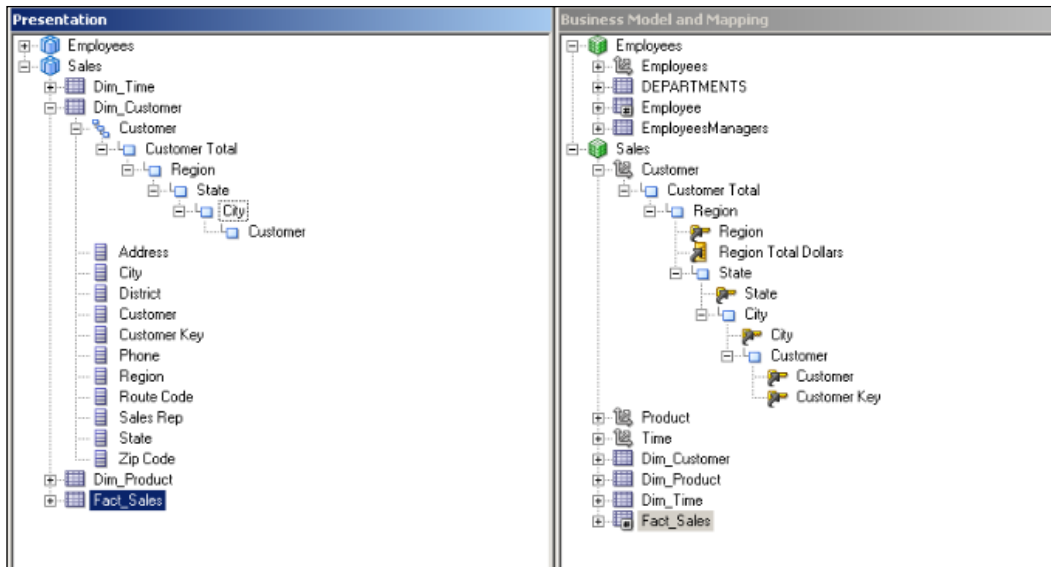
2. We'll repeat the same task for the `Product` dimension as well. After that we'll have all the hierarchies at the Presentation layer. Now users will be able to use the hierarchies in their analyses.
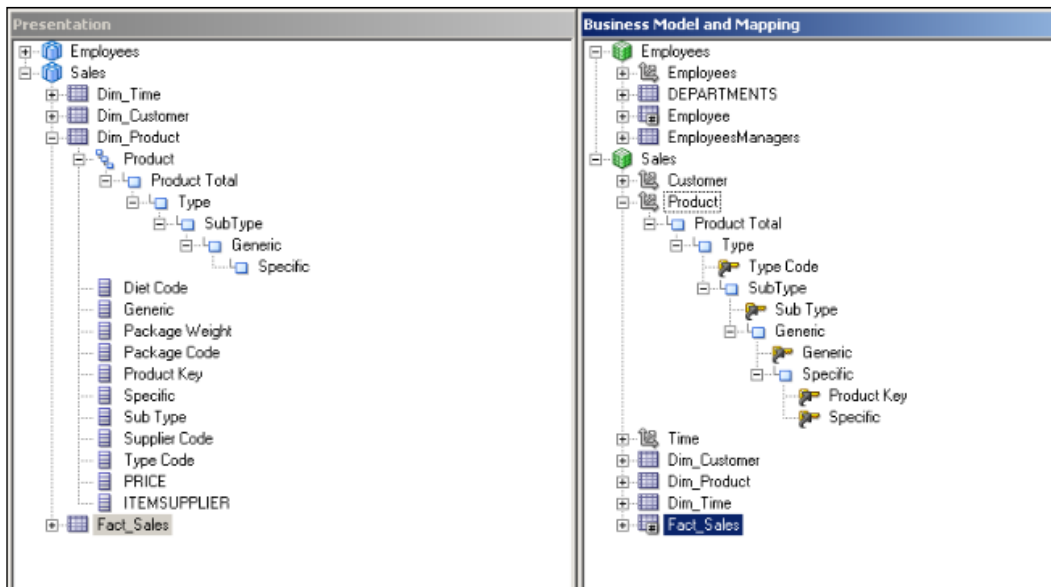
How it works...

1. Now it's time to check these hierarchies at **Presentation Services**. After logging in to Presentation Services, we're going to start creating a new **analysis**. You'll see the new hierarchies in the **Subject Area** pane. The following are the two dimensions in our sample scenario:

   ❑ You can see the `Product` dimension in the following screenshot:
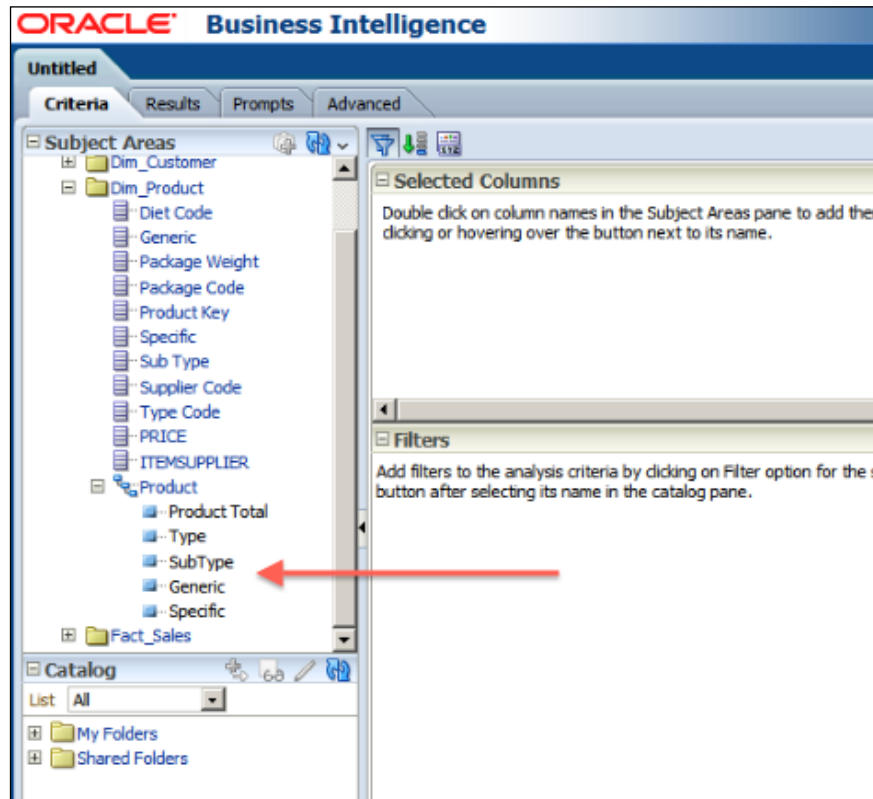
❑ You can see the `Customer` dimension and its levels in the following screenshot:

2. Now, instead of dragging any presentation column to the **Selected Columns** section, we'll drag the entire hierarchy. We're going to add a measure column as well. We're going to use the `Customer` hierarchy. So both objects are added to the **Selected Columns** area.
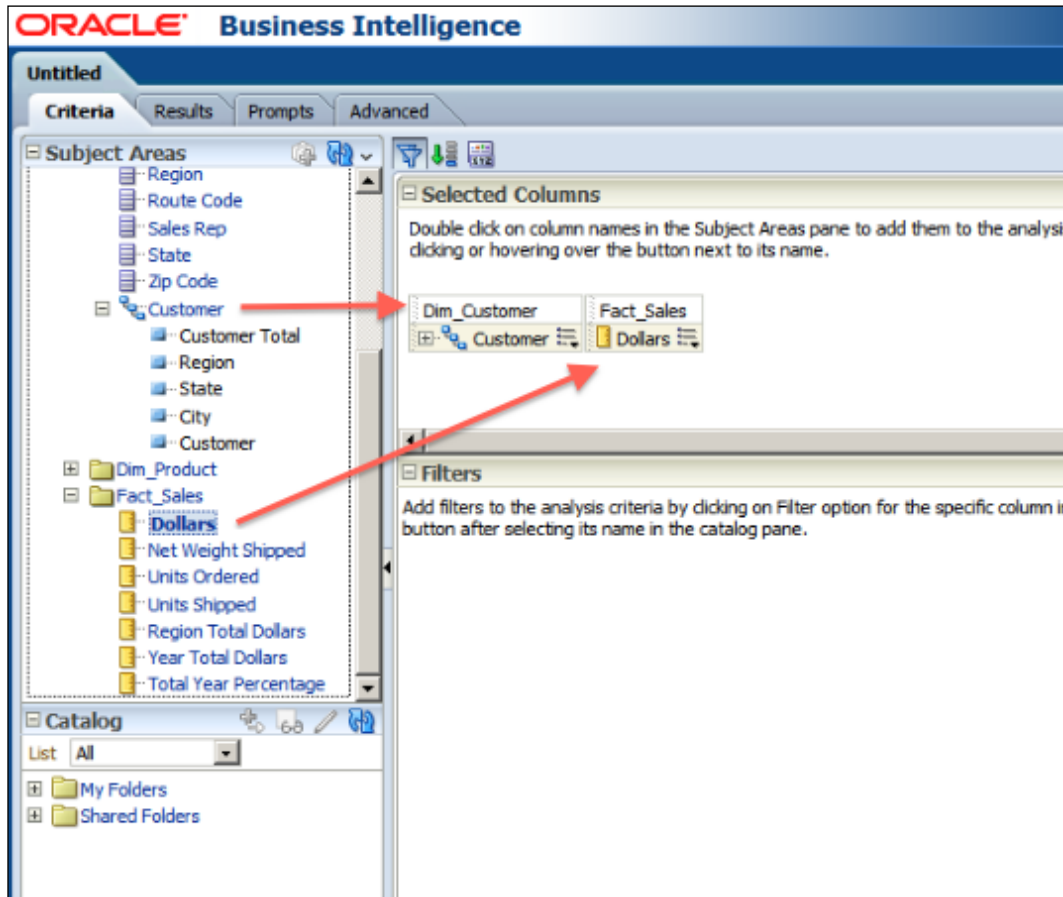
3. When you click on the **Results** tab, you'll see **that Pivot Table** view is added to the analysis because of the hierarchy. If you have used a presentation column, a **Table** view was going to be added to the hierarchy. This is the default behavior in the **OBI 11***g* version. End users will expand and collapse the levels easily with this **Pivot Table** view.

There's more...

If you don't want to use hierarchies in the **Criteria** tab in the Analysis Editor, then the **Table** view is going to be added. And this time, users are going to gain the benefit of the drill-down feature.

You'll see that only the **Region** presentation column and the **Dollars** measure are added to the analysis.
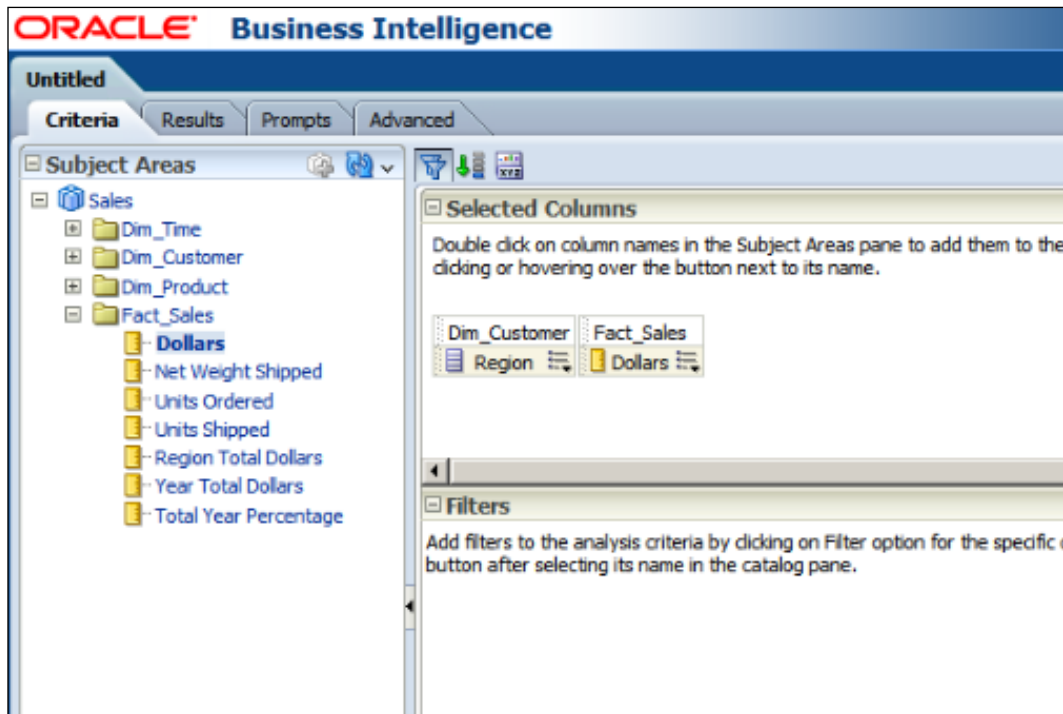
Now the users may drill down by clicking on the hyperlink easily.
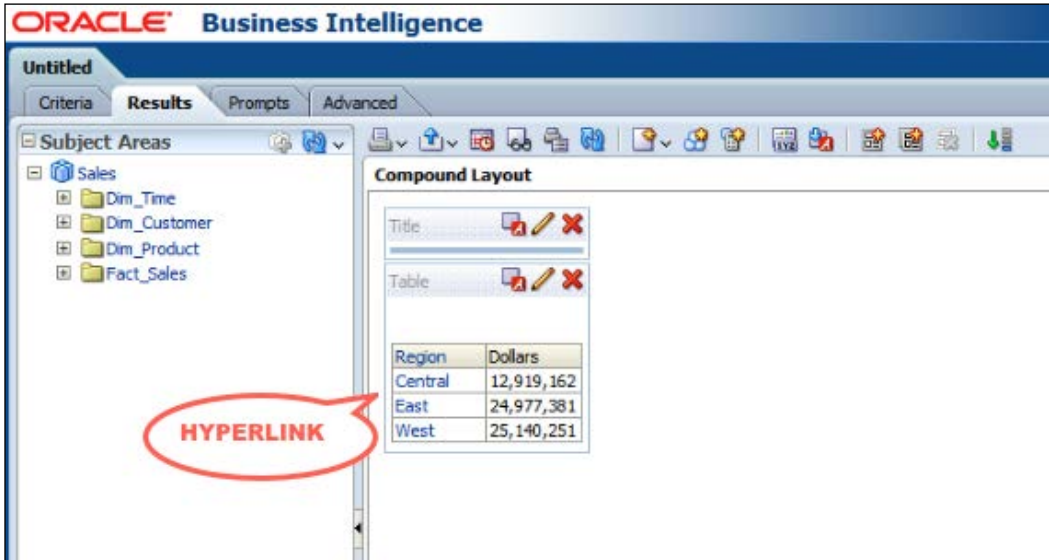
Clicking on the hyperlink will navigate you to one lower detail. In our case, the **West** value is clicked and it drills down to details.

3
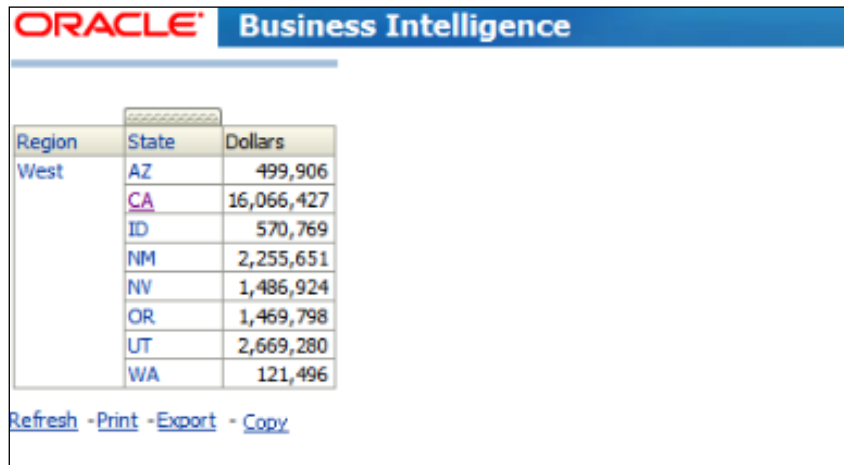
Using Aggregates and the Time Series Functions

In this chapter, we will cover:

- ▸ Creating aggregate tables
- ▸ Implementing aggregate tables
- ▸ Using the Aggregate Persistence Wizard
- ▸ Creating calculations by using the time series functions

Introduction

As a general practice, the data warehouse is maintained at the lowest granular level. Data warehouse performance bottlenecks are often due to **measure aggregation**. For example, to have some amount of dollars at different levels of dimension hierarchy, a calculation will be needed at runtime. This will impact the performance. Business users should wait for the result set until the calculation regarding the required aggregation is done. Based on the amount of data, the calculations at runtime will be very resource intensive.

So in order to improve the performance of the queries, we're going to use aggregate tables (summary tables). Aggregate tables store precomputed measure values that have been aggregated at different levels of hierarchies. These tables will make the queries run faster. After having aggregate tables, the queries won't consume as much hardware resources as before.
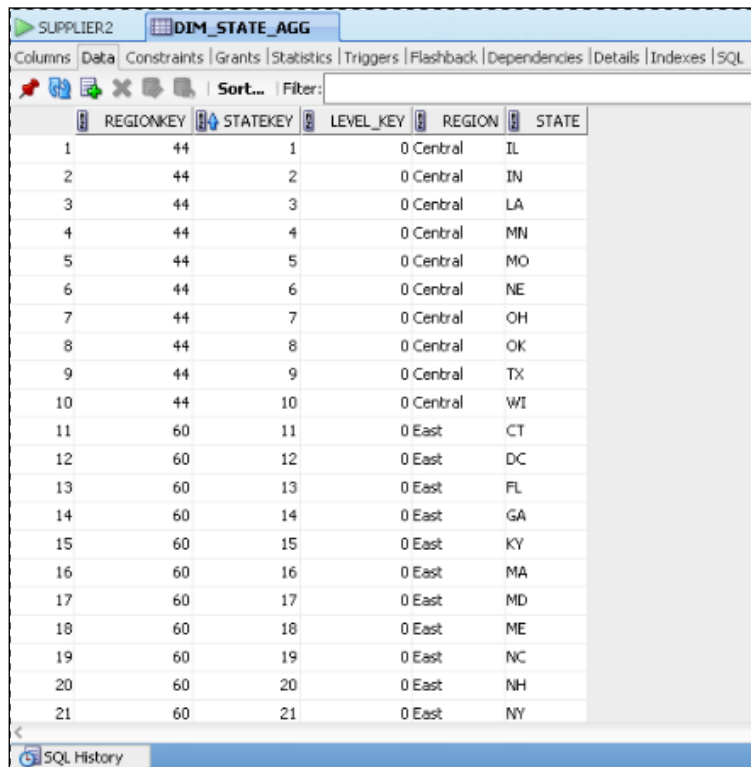
Creating aggregate tables

Aggregate tables should be designed according to business requirements. We should monitor the usage statistics of the analysis and then we'll make a decision about the levels of hierarchies that will be stored in the aggregate tables.

How to do it...

1. Let's assume that the most frequently accessed hierarchy levels are:

   ❑ **Product hierarchy**: Subtype level

   ❑ **Time hierarchy**: Year level

   ❑ **Customer hierarchy**: State level

   So we're going to create three aggregate dimension tables and one aggregate fact table based on our sample scenario. The first table is named as DIM_STATE_AGG. This table stores the attribute columns regarding the STATE level and also it contains higher levels as well, which is the REGION level. You can see the sample view of the table in the following screenshot:

2. The second table stores `SUBTYPE` and `TYPE` levels of data and it's called as `DIM_SUBTYPE_AGG`.

	TYPEKEY	SUBTYPEKEY	LEVELKEY	SUBTYPE	TYPE_CODE
1	178	1	0	4000	105
2	318	2	0	4050	107
3	305	3	0	4100	102
4	303	4	0	4150	106
5	284	5	0	4200	114
6	269	6	0	4250	120
7	307	7	0	4300	100
8	232	8	0	4350	110
9	307	9	0	4400	100
10	309	10	0	4450	101
11	292	11	0	4500	118
12	307	12	0	4550	100
13	303	13	0	4600	106
14	307	14	0	4650	100
15	303	15	0	4675	106
16	307	16	0	4700	100
17	284	17	0	4750	114
18	309	18	0	4800	101
19	178	19	0	4850	105
20	312	20	0	4900	112
21	306	21	0	4950	115

3. The last aggregated dimension table is `DIM_YEAR_AGG`. It only stores `YEAR` level of data. The parent level of `YEAR` level is `GRAND TOTAL`. So there are no other levels that are higher than `YEAR` level.

	YEARKEY	LEVELKEY	YEAR
1	1	0	1998
2	2	0	1999

4. Here is the aggregated fact table, which is called as `FACT_SALES_AGG`. This table stores precomputed measure that are aggregated at `YEAR`, `SUBTYPE`, and `STATE` levels.

	STATE	SUBTYPE	YEAR	DOLLARS	NET_WEIGHT	UNITS_ORDERED	UNITS_SHIPPED
1	19	92	1	27352.109931468963495	21341.5299444198607469	2292	2268
2	19	101	1	6959.0600309371948055	6076.20000076293945	307	307
3	19	29	1	97871.11990165710435	20048.340136408805783	3649	3627
4	19	65	1	22320.020111083984273	11817.479978561401319	503	501
5	19	18	1	22578.130045890808045	10098.28999328613281	976	965
6	4	138	1	2132.3899788856506393	1164.089994430541983	134	129
7	5	39	1	2155.5699977874755786	1250.519995480775832568	329	320
8	5	55	1	34056.880115509033069	11658.199970245361282	664	608
9	5	5	1	27094.010049819946231	9509.5699965953826913	623	617
10	5	16	1	660.6699972152709951	166.07000100612640414	46	44
11	5	142	1	24081.4699262380599763	14239.96998041868207731	883	789
12	28	75	1	408979.7808383703226888	428218.3301284313198794	22488	22249
13	28	42	1	127543.21974870562533805	72471.6100335121153272	4477	4327
14	28	6	1	79788.4100046157833886	161441.74018669128411	5659	5509
15	28	101	1	10142.2900376319885439	8932.9200141429900952	673	647
16	28	1	1	1034087.7109723091121026	569561.918435573577542	27165	26888
17	28	146	1	33454.3500020503996894	11966.650019645690896	1905	1870
18	28	76	1	103744.2001163959499669	90128.6501698493954937	11381	11280
19	28	136	1	2308.38999989628791345	500.79000209271907851	76	71
20	28	78	1	307617.64095115661578	186282.96003007888789694	7559	7339
21	28	98	1	159765.989922888576614...	232303.839965820312401	10031	9795

How it works...

After creating these aggregate tables, we'll expect that whenever someone executes a query that includes the sum of `DOLLARS` and `STATE` attribute column, the query will be satisfied from the `FACT_SALES_AGG` table instead of the detailed fact table. Eventually the query response time will be reduced and it won't be a resource intensive query. At the end, this will improve the performance of the queries. We'll have to make some changes in the repository so that the BI server can navigate the query to the proper fact table depending on the selected levels.

There's more...

Creating and maintaining these aggregate tables also has a cost. Normally, it's not easy to modify the database, because creating one more table means that new **Extraction, Transformation, and Loading** (**ETL**) **processes** need to be defined and maintained, in addition to the ones that already exist.

Plus, there could be many aggregate tables depending on the number of dimensions and the number of levels in the hierarchies. Creating all of the aggregate tables for every combination of levels at different dimension hierarchies is also not recommended. It won't be easy to maintain the ETL process. On the other hand, having aggregate tables will definitely improve the performance. So we should balance the number of aggregate tables. We should first monitor the usage statistics, and then we should decide about the levels that the aggregate tables would store.

Let's assume that the data warehouse is populated every night at 2:00 A.M. Now once the detailed fact table is loaded from various sources, you'll have to recalculate these aggregate tables. Having many aggregate tables will increase the time of the ETL process.

Using a multidimensional source is another solution for summary reports. Instead of creating aggregate tables, we can use the Essbase server to create **OLAP Cubes** and store the summary results in these cubes. This is going to be covered in the *Chapter 4, Working with Multidimensional Data Sources*.

Implementing aggregate tables

In order to improve query performance, we have created aggregate tables at the database level. Now we're going to modify the Physical and BMM layers of the repository to gain benefit of summary data.

- ▶ **Physical layer**: Aggregate tables should be imported and physical joins should be created

- ▶ **Business Model and Mapping layer**: Secondary logical table sources are going to be added to the corresponding logical tables

- ▶ **Presentation layer**: No modification will be made at this layer. Usage of aggregate tables will be transparent to the end users

How to do it...

1. As we have already demonstrated how to import tables into the Physical layer in *Chapter 1, Exploring and Building the Repository*, you won't find every detailed step in this task. Here is the screenshot that shows the imported tables.
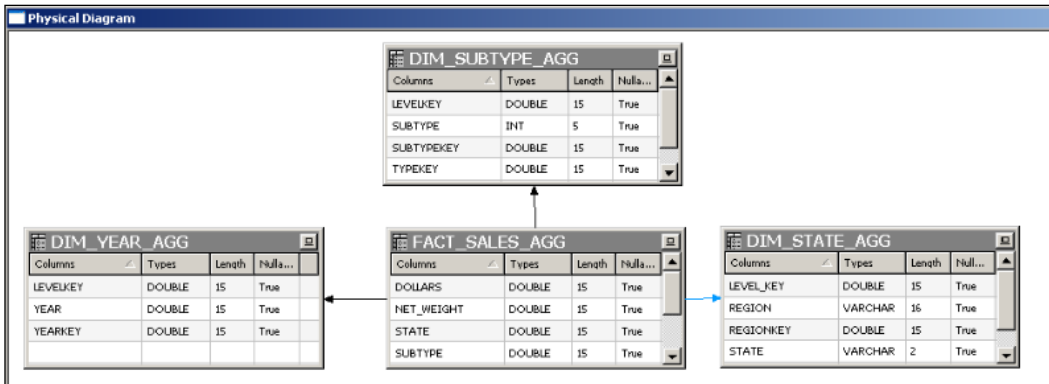
2. The physical joins between the fact table and all three dimensions tables should be created. The first physical join is between `FACT_SALES_AGG` and `DIM_SUBTYPE_AGG` tables. The join condition is as follows:

```
"orcl".""."SUPPLIER2"."DIM_SUBTYPE_AGG"."SUBTYPEKEY" =
"orcl".""."SUPPLIER2"."FACT_SALES_AGG"."SUBTYPE"
```

The definition of the join object is as seen in the following screenshot:
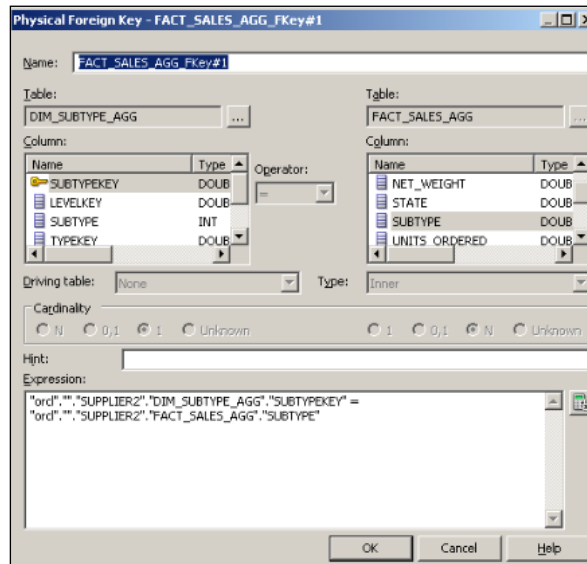
3. The second join is between `FACT_SALES_AGG` and `DIM_STATE_AGG` tables and here is the join condition:

    ```
    "orcl"."".."SUPPLIER2"."DIM_STATE_AGG"."STATEKEY" =
    "orcl"."".."SUPPLIER2"."FACT_SALES_AGG"."STATE"
    ```

    The definition of the join object is as seen in the following screenshot:
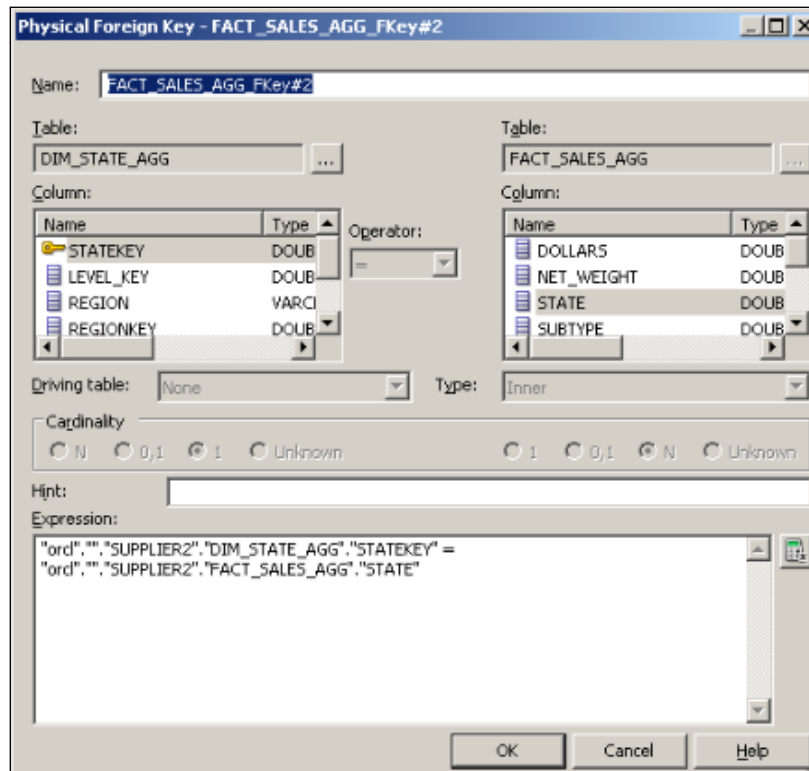
4. The last join is between `FACT_SALES_AGG` and `DIM_YEAR_AGG` tables and its condition is as follows:

    ```
    "orcl"."".."SUPPLIER2"."DIM_YEAR_AGG"."YEARKEY" =
    "orcl"."".."SUPPLIER2"."FACT_SALES_AGG"."YEAR"
    ```

The definition of the join object is as seen in the following screenshot:
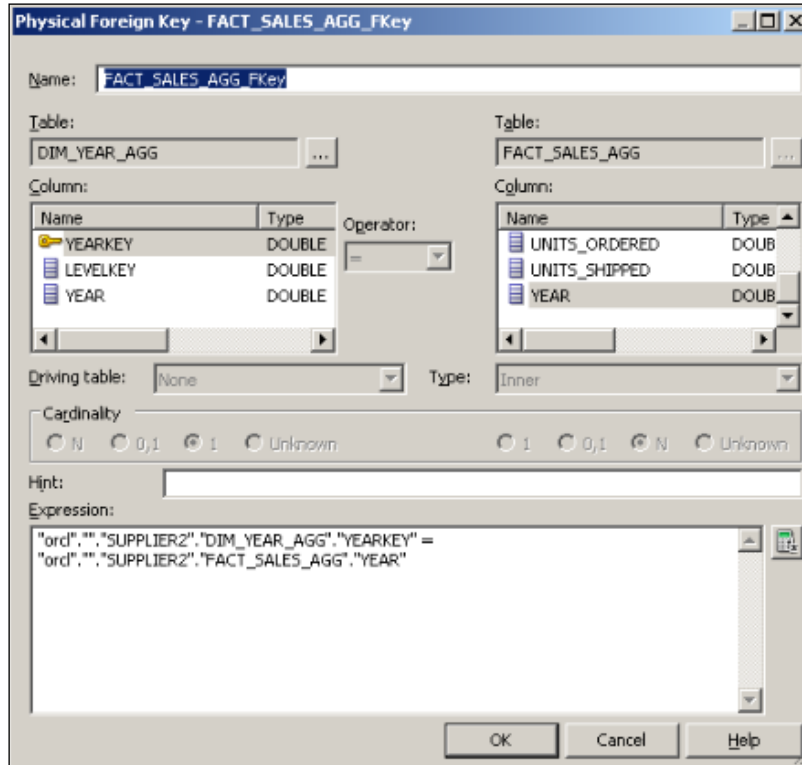
5. Now we'll have to make modifications at the Business Model and Mapping layer. We will add one more logical table source to the `Dim_Customer` logical table. The first logical table source will be mapped to the detailed dimension table at the Physical layer. The second logical table source will be mapped to the aggregate dimension table.

6. Create a new logical table source by right-clicking on the `Dim_Customer` logical table and selecting the **New Object** | **Logical Table Source** option. This will bring up the new window and we're going to set the name of the logical table source as `Agg_CustomerLTS`.

7. We're going to select the physical table by clicking the **+** sign. After selecting the aggregate dimension table, you'll see the window as displayed in the following screenshot.
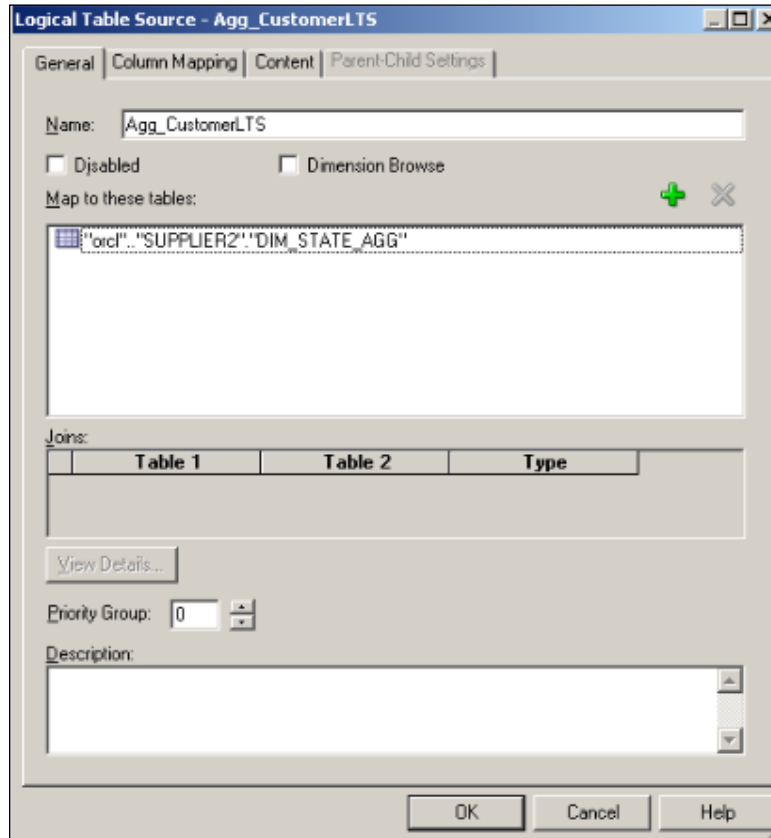
8. Now we're going to make the column mapping on the **Column Mapping** tab of the same **Logical Table Source**. You'll see that only two columns are mapped and all the others are unmapped. This is because the aggregate dimension table doesn't store any details except the `STATE` and `REGION` levels.
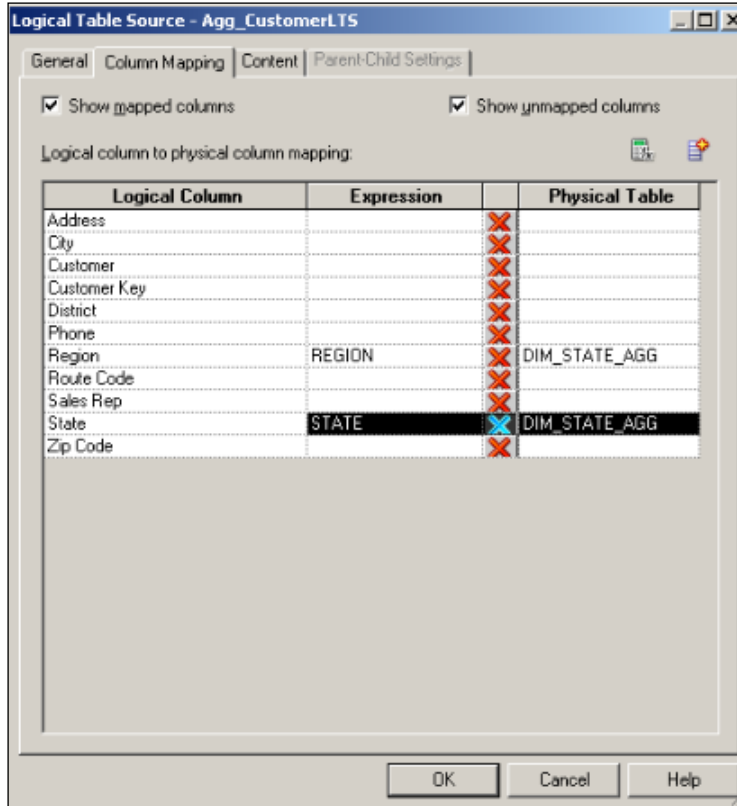
9. The last modification will be done on the **Content** tab of the **Logical Table Source**. We'll have to define the levels at which this logical table source can satisfy queries. This step is very important. Making any kind of mistake will cause the BI server to use a wrong logical table source for the queries. We're going to select the `State` level for the `Customer` **Logical Dimension**. Setting this property means that any query that contains the State or higher level columns (such as Region Level) will be satisfied by using the `AggCustomerLTS` logical table source.
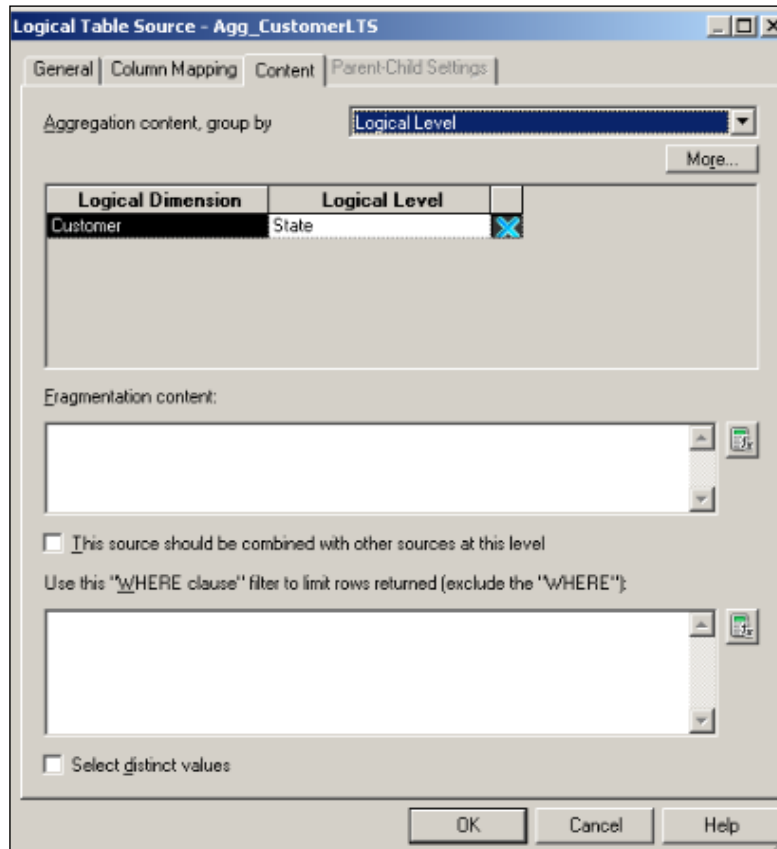
10. The Content tab of the Logical Table Source window shows the levels that this source would satisfy. So we have to modify the content of the first logical table source to the most detailed level in the hierarchy.
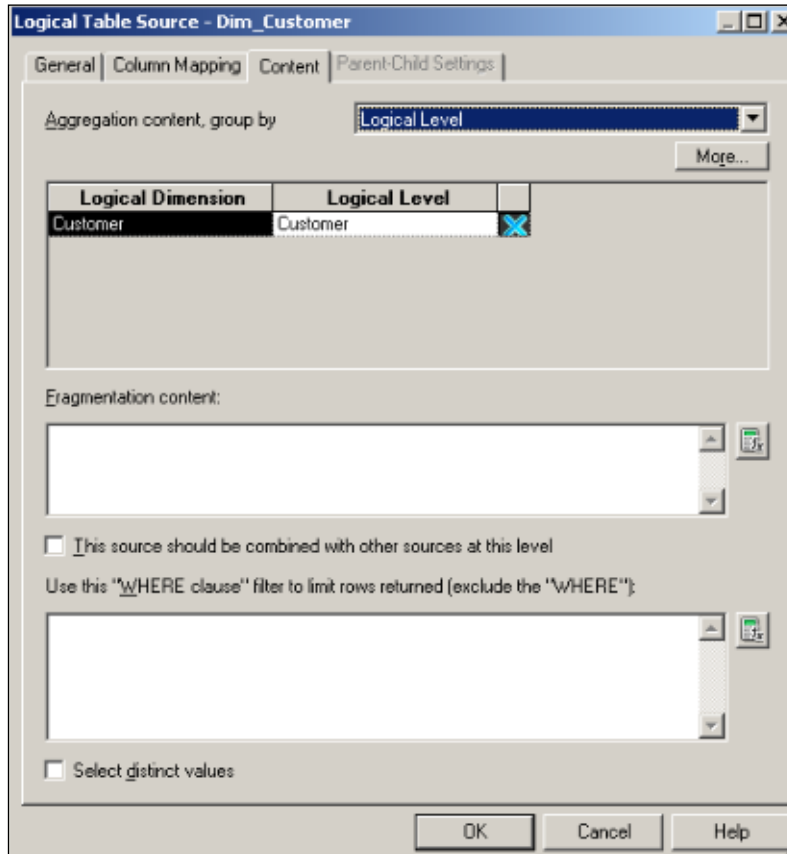
11. We have created the second logical table source for the `Dim_Customer` logical table. Expand the `Sources` folder to see the logical table sources. The BMM Layer will look like the following screenshot:
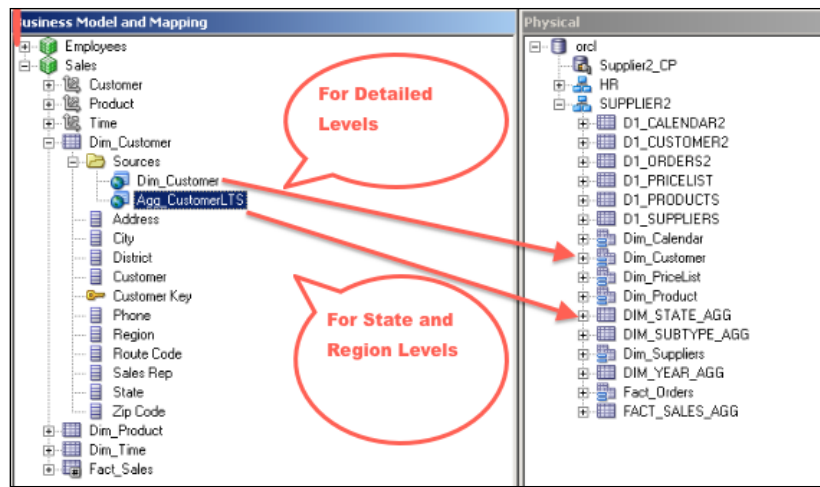
12. It's time to do similar steps for the remaining two dimension tables and one fact table. The steps are going to be exactly the same. We're going to add the second logical table source that will be mapped to the proper aggregate dimension table and then check the column mappings in order to be sure about whether the correct mappings are done. The last step will be about setting the proper level depending on the data that is stored in the aggregate tables. So you'll see that the second logical table source of the `Dim_Product` table is also created as follows:
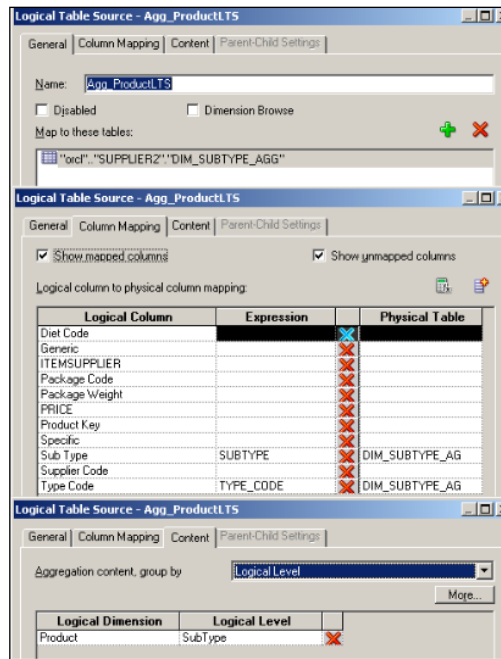
13. The details of the logical table source that belongs to the `Dim_Time` logical table are as follows:
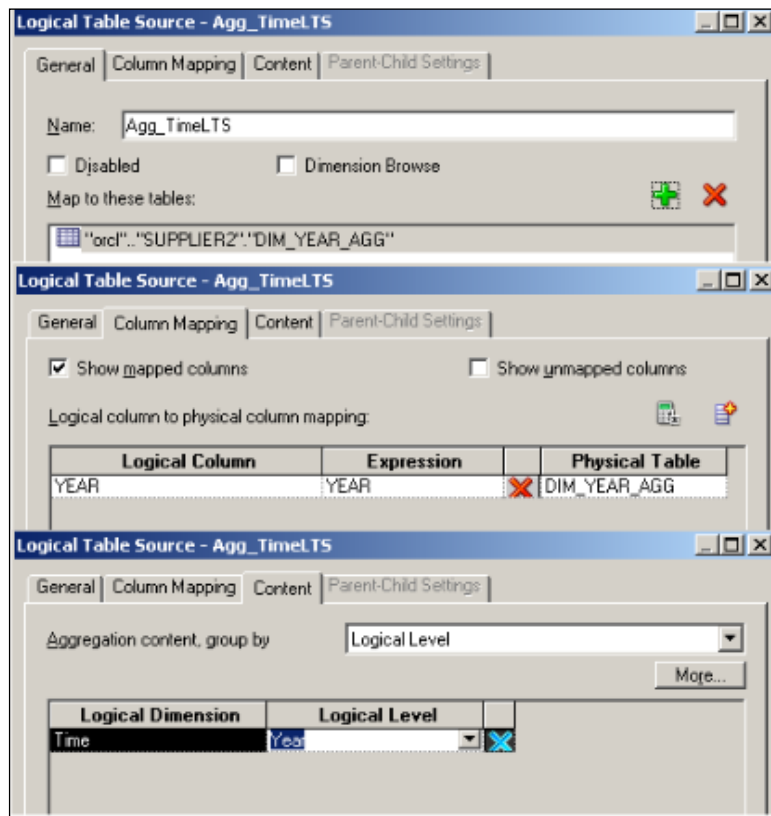
14. Here are the details of the last logical table source that belongs to the fact table.

How it works...

There won't be any modification at the Presentation layer. Business users are going to select the columns in the analyses as they work. Using aggregate tables is transparent to the users. In the background, the BI server is going to navigate the request to the proper logical table source at the BMM layer according to the definition of the logical table sources. For example, we're going to create an analysis that includes only two columns. The selected columns are `YEAR` and `Dollars`.
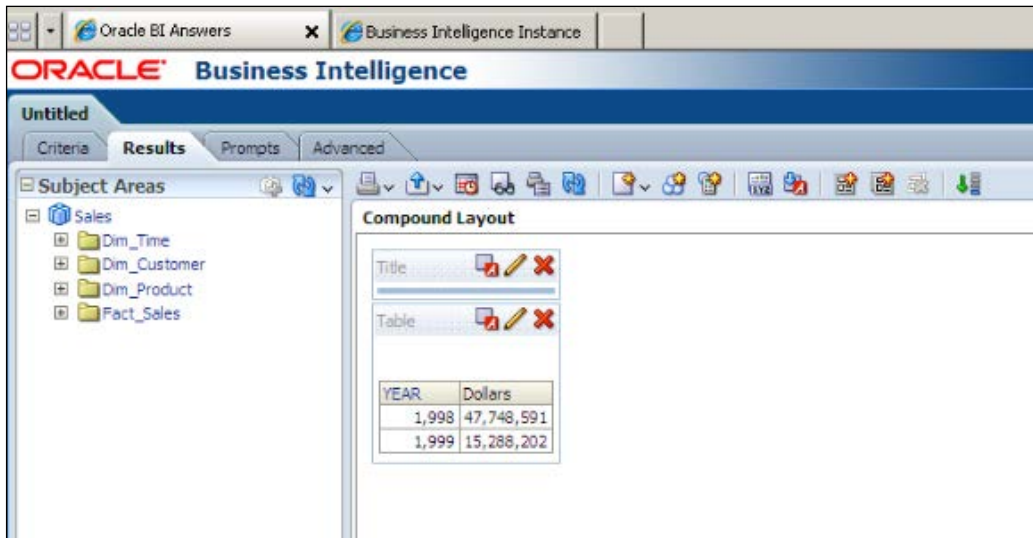
Because the BI server is aware of the aggregated tables it's going to access the aggregate tables instead of accessing the detailed fact table. There won't be any calculation at runtime. It's going to just retrieve the result set from the precomputed aggregate table. It will be useful to enable the query logging at the user level. You can increase the user's logging level from `0` to `2` in the Identity Manager that is part of the BI Administration Tool. Once the logging level is set to `2`, the physical SQL statements are going to be logged into a file named `NQQuery.log`. BI developers won't be able to access this file directly because it's stored in the file system on the BI server. Developers can access the content of the logfile from Presentation Services. After logging into Presentation Services, navigate to **Administration** | **Manage Sessions**.

The content of the log is as follows:

```
WITH
SAWITH0 AS (select sum(T946.DOLLARS) as c1,
  T942.YEAR as c2
From
  DIM_YEAR_AGG T942,
  FACT_SALES_AGG T946
where ( T942.YEARKEY = T946.YEAR )
group by T942.YEAR)
select distinct 0 as c1,
  D1.c2 as c2,
  D1.c1 as c3
from
SAWITH0 D1
 Order by c2
```

There's more...

There's also a cost of using aggregate tables besides their advantages. These aggregate tables should be maintained during the ETL process, which is a time-consuming task. We won't be able to create many aggregate tables because of the time pressure. We should create the aggregate tables after finding out which levels are frequently accessed. As the data warehouses will be refreshed during the non-business hours, it's a limited duration. Let's assume that refreshing of the data warehouse is going to be started at midnight. Obviously, this entire process should finish before the business hours start. First, the detailed fact tables and dimension tables are going to be updated, and then the summary (aggregate) tables. Modifying the aggregate tables will take time. There will be calculations and only summary data is going to be inserted into these tables. Then it'll come to do the maintenance tasks at the database level, such as gathering optimizer statistics. As you can see, there are many tasks to do starting from midnight to the beginning of the business hours. Unfortunately, we won't be able to create the summary tables according to every available perspective. We'll need to make a decision about the summary tables that will be populated. In order to make a good decision, we can enable usage tracking to monitor the end users' behavior. Enabling usage tracking will be covered in *Chapter 6*, *Managing Usage Tracking and Enabling the Cache*.

Using the Aggregate Persistence Wizard

Designing and creating the aggregate tables is not an easy task. Especially, creating the aggregate dimension tables and aggregating the fact tables will take some time. You'll have to think about the joins as well.

There's a utility called **Aggregate Persistence** in **BI Administration Tool**. This utility will help us create the aggregate tables and load the precomputed data into them automatically. But you'll have to reflect all changes to the ETL process. Once this wizard is executed, the aggregate tables are going to be created in the database. Also, summary data will be calculated and it's going to be inserted into the aggregate tables. Obviously these tables should be imported into the BI repository and the physical joins should be created. These two steps are going to be done automatically too. The next step is reflecting the changes in the BMM Layer. Actually, Aggregate Persistence Wizard will do this task too. As a summary, it's a completely automated process. The new physical tables are going to be named based on a setting in the `NQSConfig.INI` file. This file is the configuration file of the BI server. The default value of the `AGGREGATE_PREFIX` parameter is `SA_`. So all the new tables will have a prefix as `SA_`.

In order to demonstrate the usage of this utility, let's assume that these are the most frequently accessed levels:

- ▶ **Time dimension**: Month level
- ▶ **Product dimension**: Subtype level
- ▶ **Customer dimension**: Region level

At the end of the wizard, you'll see that these levels are created and modified automatically.

Aggregate tables are created and precomputed data is inserted into these tables. They are also imported to the Physical layer. Physical joins are also automatically created. New logical table sources are added. The column mappings are done. Also all the corresponding levels are selected in the Content tab of the new logical table sources.

How to do it...

1. Open **BI Administration Tool** and open the repository. Click on the **Utilites...** option in the **Tools** menu.
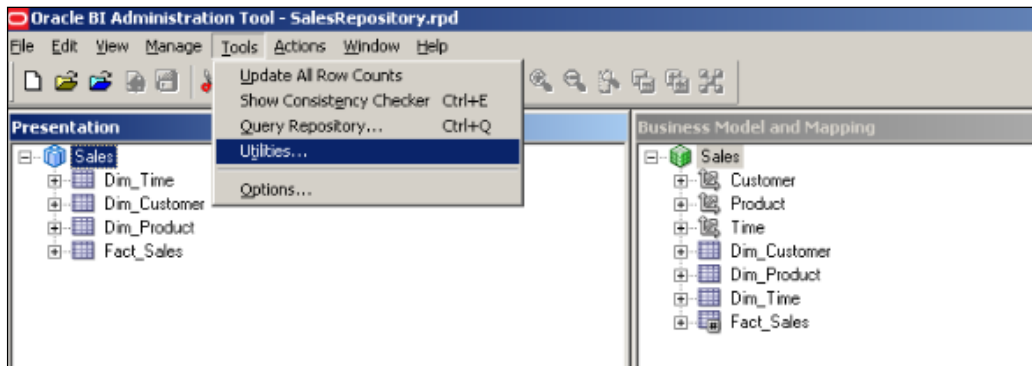
2. Clicking on **Utilities** will pop up the **Utilities** window. We're going to select the **Aggregate Persistence** option from the list and click on **Execute**.
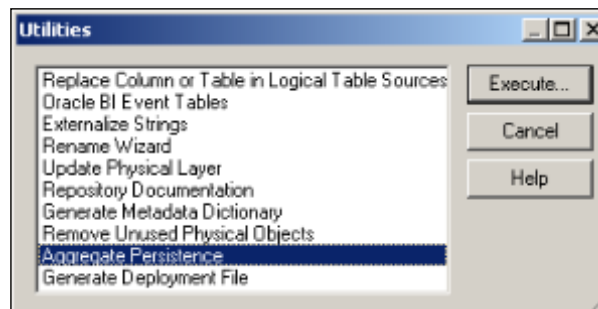
3. The Aggregate Persistence Wizard will pop up with six steps. The first step is **Select File Location** and we're going to define the name of the script file and the path of the file.
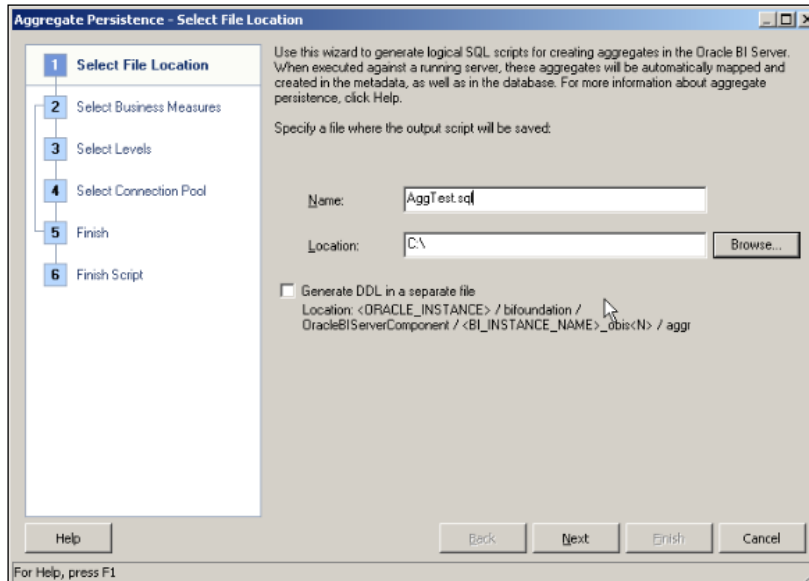
4. Select the measure that will be aggregated from the `Fact_Sales` table at the **Select Business Measures** step.
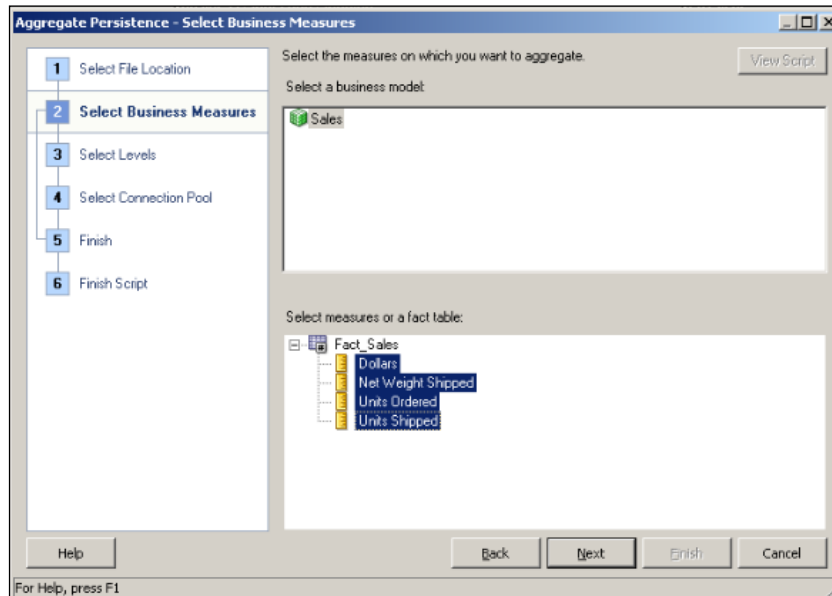
5. According to our sample scenario, we're going to the select levels of the hierarchies in order to make the aggregation in the **Select Levels** step. We can also select the **Use Surrogate Key** checkboxes to define them on the new aggregate tables. Surrogate keys are used to distinguish the records from each other during the ETL process. We're not going to use them in our scenario.

❑ `Region` Level for the `Customer` dimension

❑ `SubType` Level for the `Product` dimension

❑ `Month` Level for the `Time` dimension
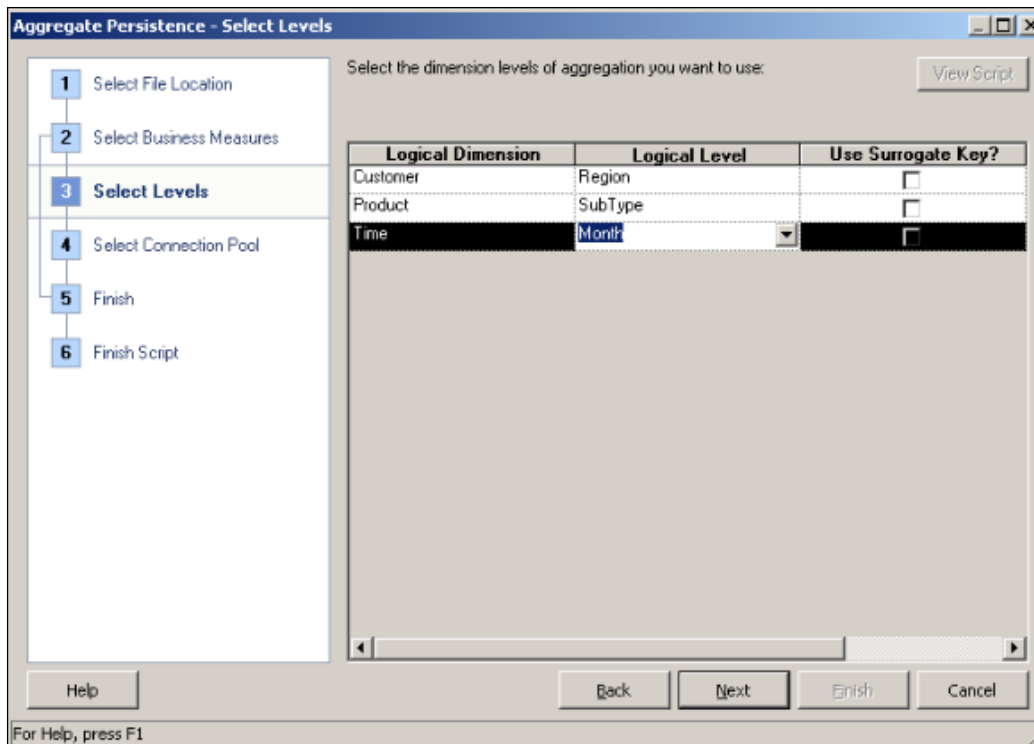
6. Now we're going to define the physical database settings in order to create the aggregate tables.

  ❑ **Database**: ORCL

  ❑ **Schema**: SUPPLIER2

  ❑ **Connection Pool**: Supplier2_CP

  ❑ **Aggregate Table Name**: ag_Fact_Sales
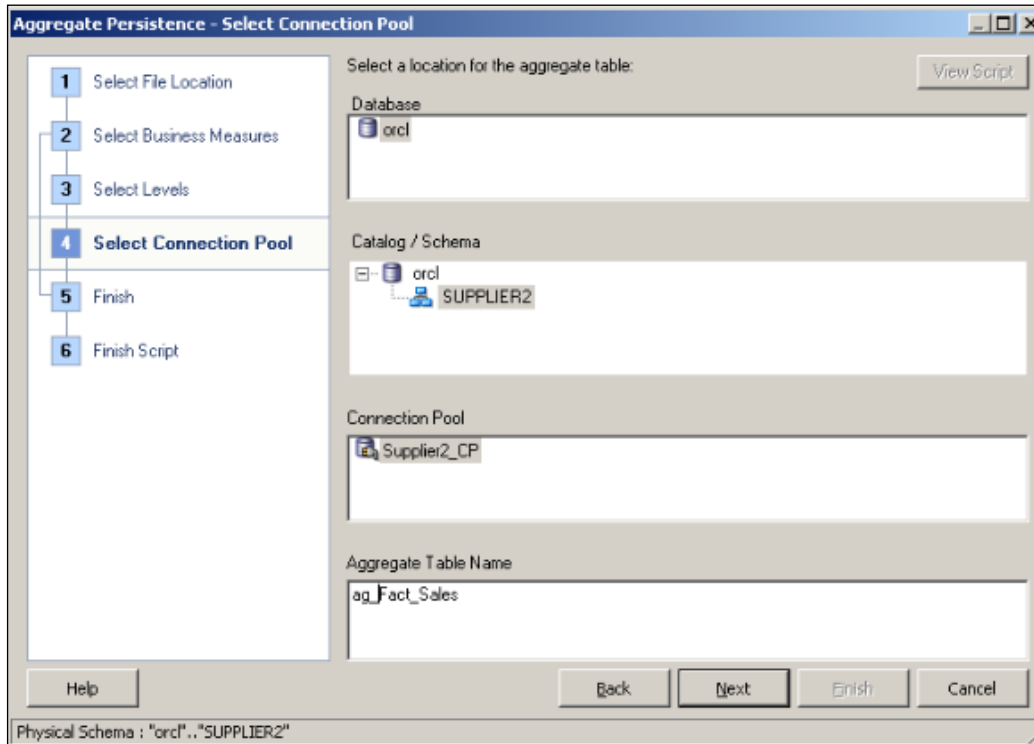
7. You'll see the preview of the script and if you're interested in defining a new
    aggregate, you may select the **Define another aggregate** option. In our scenario,
    we're going to select the **I am done** option. By clicking the **Next** button, it'll take us
    to the last step.
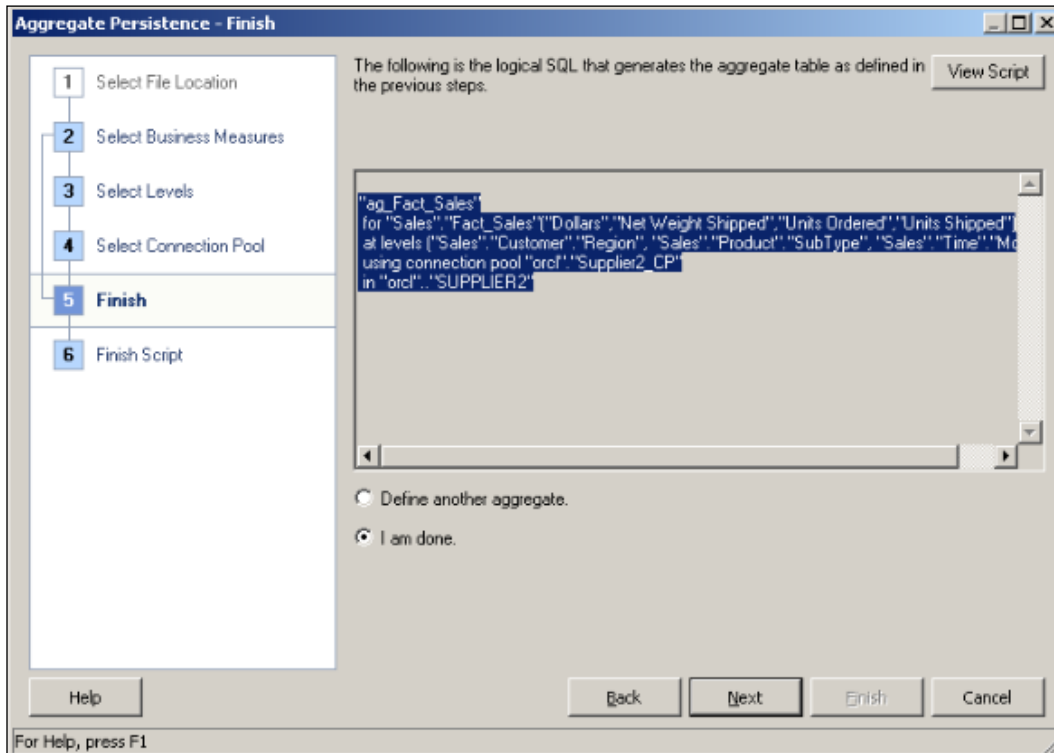
8. The last step of the wizard is the summary screen. It shows the path of the file that is created and the content of the script. We'll click on **Finish**.
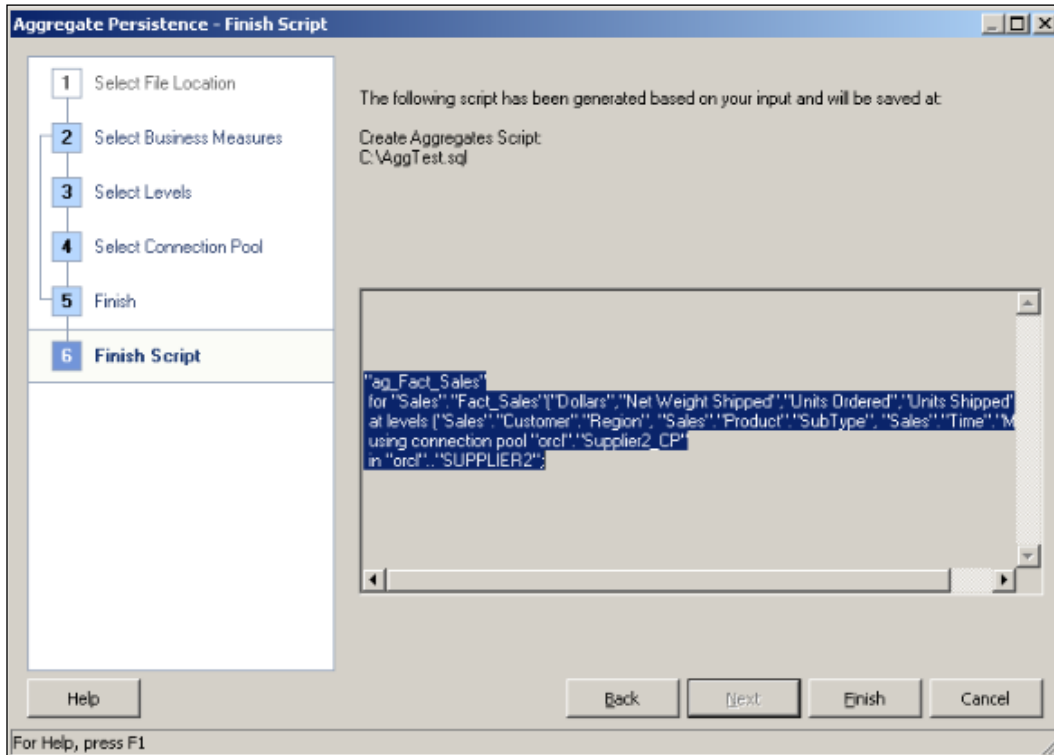
9. When you check the `C:` location, you'll see that `AggTest.sql` file is created. You can see the content of the file.
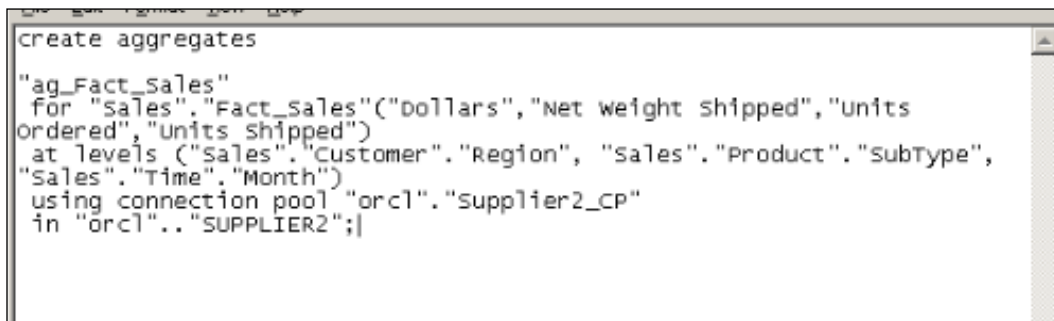
10. Now it comes to executing this BI server script. There are three ways to execute it:

   ❑ **Command Line**: By using `nqcmd.exe`

   ❑ **Job Manager**: By creating a new job that will use the `nqcmd` script

   ❑ **Presentation Services**: By executing the script from Presentation Services.

   We're going to use the third option in the demonstration. After logging into Presentation Services, you'll see a link named **Administration** above the global toolbar.
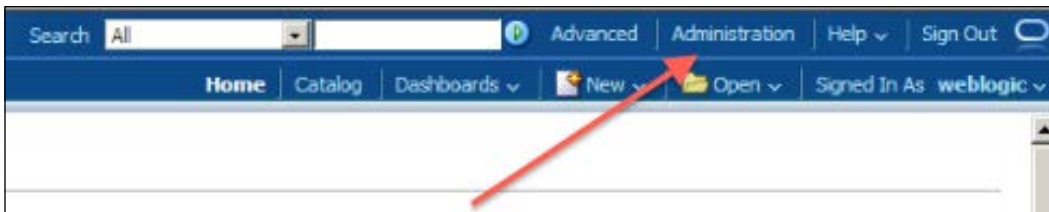
11. Clicking on **Administration** will open the **Administration** page. When you scroll down, you'll see the **Issue SQL** hyperlink.
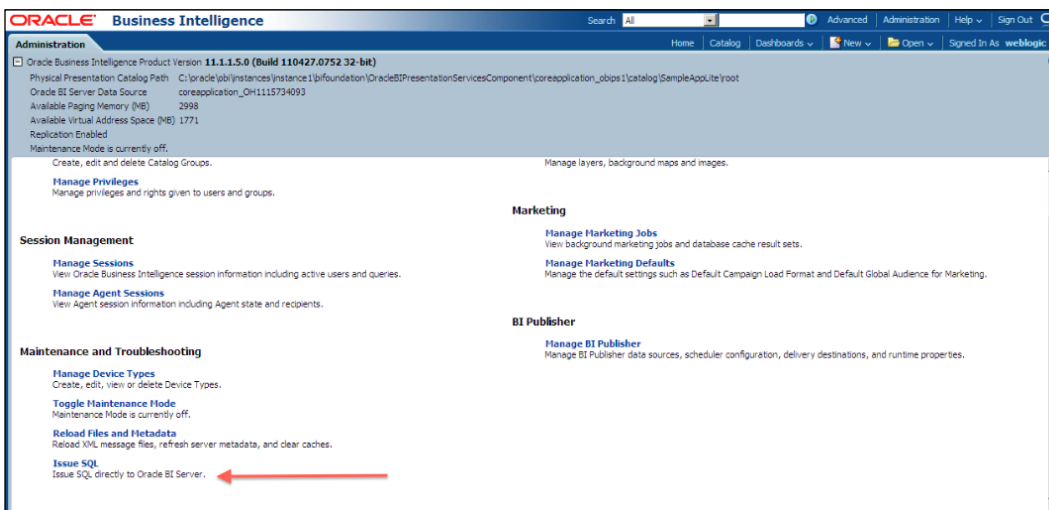
12. Click on the **Issue SQL** link and a new page will appear on the screen. Paste the contents of the `AggTest.sql` file and click on the **Issue SQL** button.
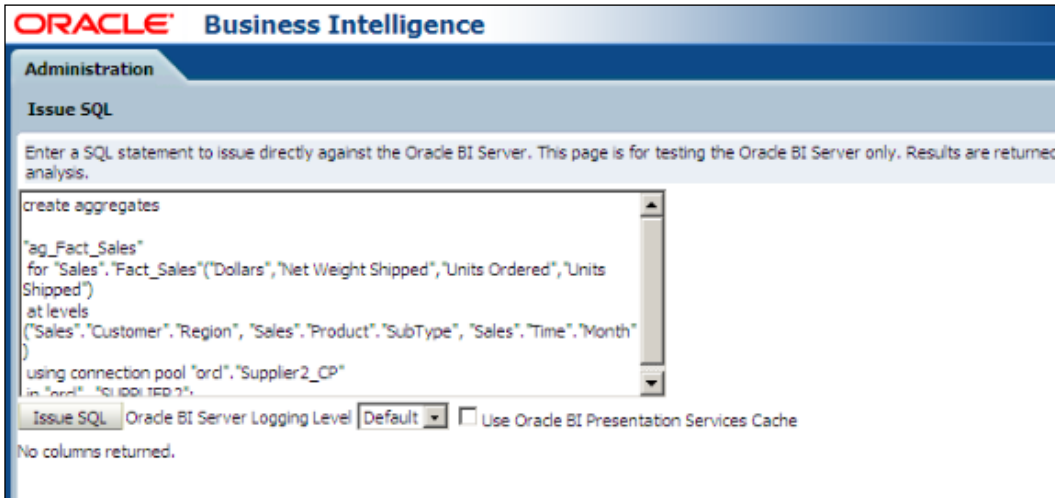
How it works...

After the SQL statement is executed, we can check the modifications in the repository. You'll see that the aggregate dimension and fact tables are created in the `Supplier2` schema; also precomputed data is inserted into these tables. Plus they're already imported to the Physical layer.
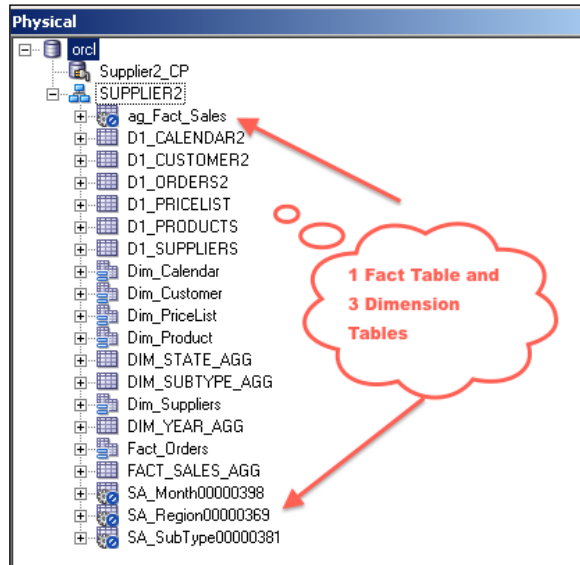
We can easily check the modifications at the BMM layer. The new logical table sources are created and all of the settings of the sources were done with the script execution.
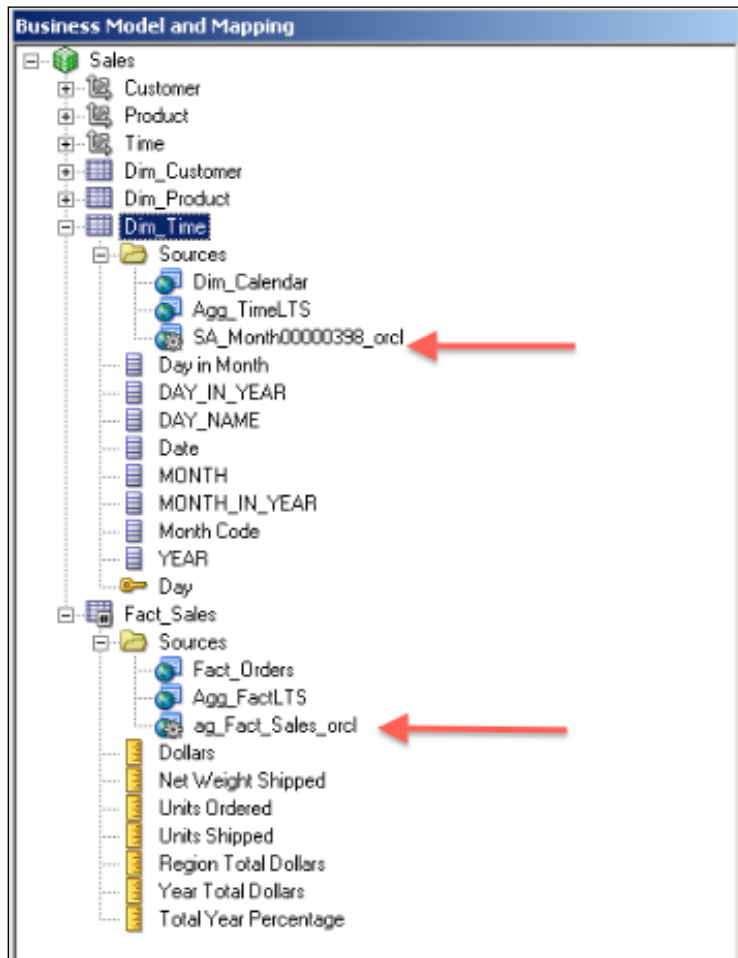
You'll also see that all the proper hierarchy levels are set up correctly for all logical dimensions.
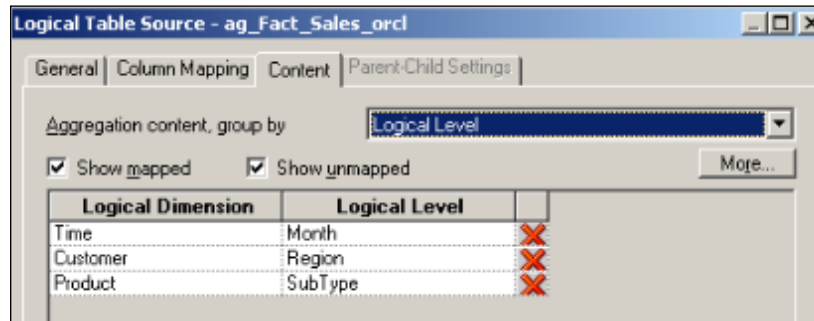
There's more...

As you see, creation of the aggregate tables is really easy by using the Aggregate Persistence Wizard. You should only consider the changes that you'll have to do in the ETL processes.

Creating calculations by using the time series functions

It's very important to enable end users to compare the business performance with previous time periods. By achieving this, end users will understand the business better. Business users need to make these comparisons across multiple time periods.

For example, let's assume that business users are asking for the sales amount of this month and the sales amount for last month. Most probably, they're looking forward to comparing these two values and seeing the difference. Here is one method:

- ▶ **Calculate the sales amount of this month**: This can be achieved by one query
- ▶ **Calculate the sales amount of last month**: This is another query
- ▶ **Calculate the difference between these months**: This will be the last query

As you see, we'll need at least three queries. There's another solution in Oracle Business Intelligence Enterprise Edition. It's time series functions. There are three types of time series functions:

- ▶ **AGO**: Calculates aggregated value as of some time period shifted from current time
  - ❑ **Syntax**: `Ago(<<Measure>>,<<Level>>,<<Number of Periods>>)`
  - ❑ **Measure**: Dollars, Units Ordered
  - ❑ **Level**: Year, Quarter, Month, Day
  - ❑ **Number of Periods**: 1, 2, 3, and so on

▶ **TODATE**: Aggregates a measure attribute from the beginning of a specified time
period to the currently displayed time

  ❑ **Syntax**: `ToDate(<<Measure>>,<<Level>>)`

  ❑ **Measure**: Dollars, Units Ordered

  ❑ **Level**: Year, Quarter, Month, Day

▶ **PERIODROLLING**: Performs an aggregation across a specified set of query
grain periods

  ❑ **Syntax**: `PeriodRolling(<<Measure>>, <<integer>>,`
   `<<integer>>)`

  ❑ **Measure**: Dollars, Units Ordered

  ❑ **Integer**: 2 (last 2 periods), 3 (last 3 periods)

  ❑ **Integer**: 0 (Current period)

In order to use the time series functions, there should be at least one Time dimension like in
the following screenshot:
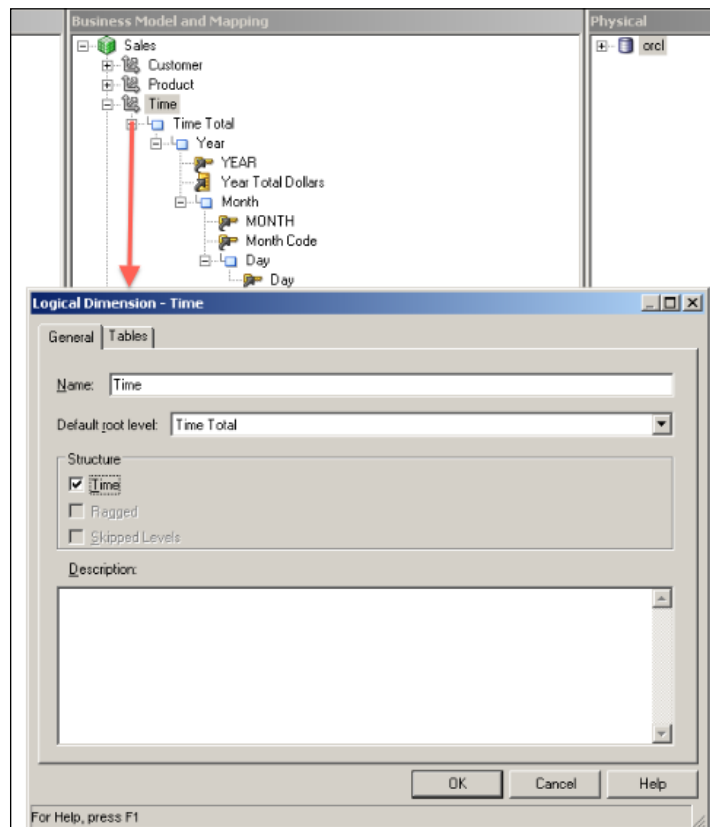
Also chronological keys should be defined at the levels of Time hierarchy. Chronological keys are used to identify the data at a particular level. The data in the time dimension needs to follow a particular order. The time series functions use these keys during the calculation.
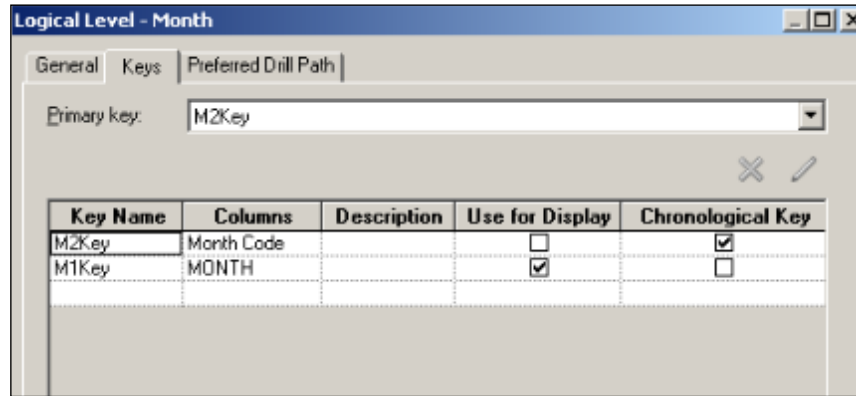
How to do it...

1. In order to use the time series functions, we're going to add a logical column to the logical table named `Fact_Sales` to display the `Month Ago Dollars` column.
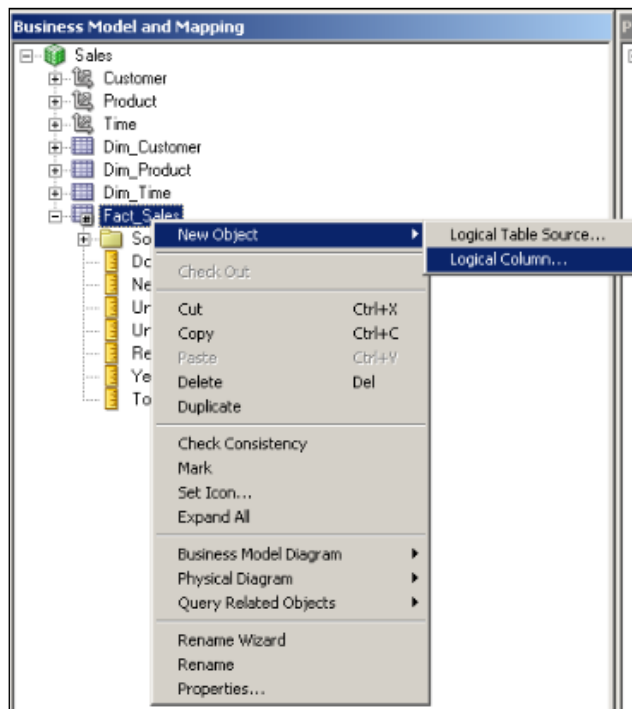
2. A new **Logical Column** window pops up. Set the name of the new column as `Month Ago Dollars`. The new logical column is not mapped to any physical column or to another existing logical column yet. We are going to define the mapping in the **Column Source** tab.
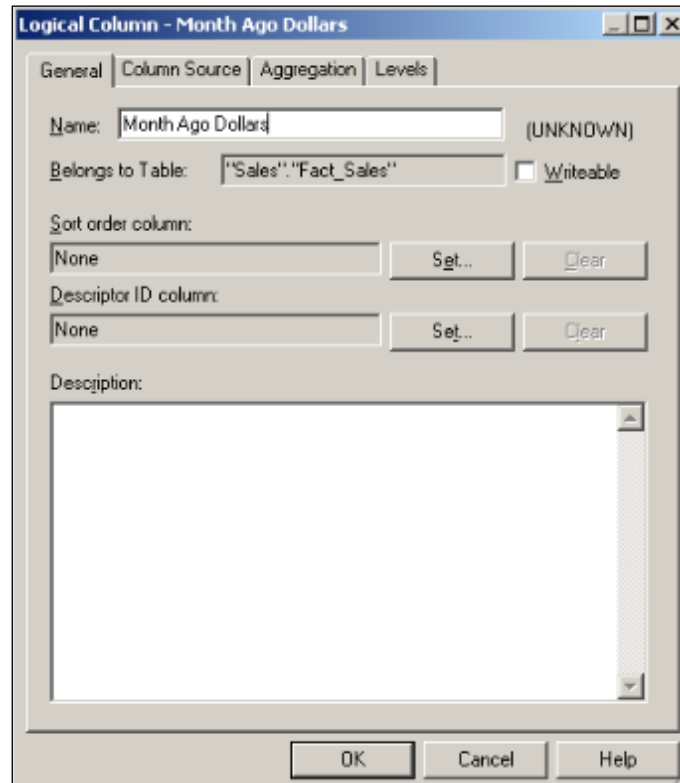
3. There are two options in the Column Source tab:

   ❑ **Derived from physical mappings**

   ❑ **Derived from existing columns using an expression**

The second option will be selected and we're going to click on the Edit Expression icon.
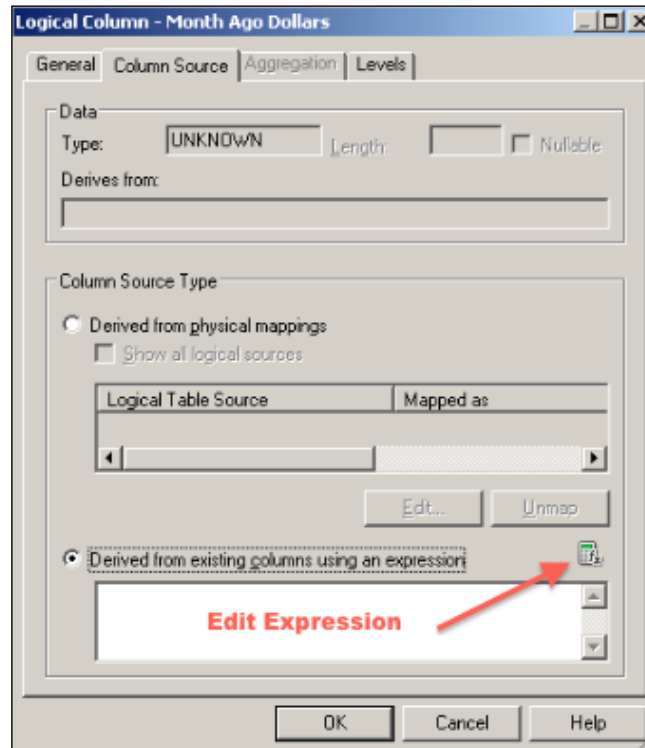
4. When the **Expression Builder** window pops up on the screen, you'll see the **Time Series Functions** option. We're going to select the `Ago` function and the list of input arguments will be displayed.
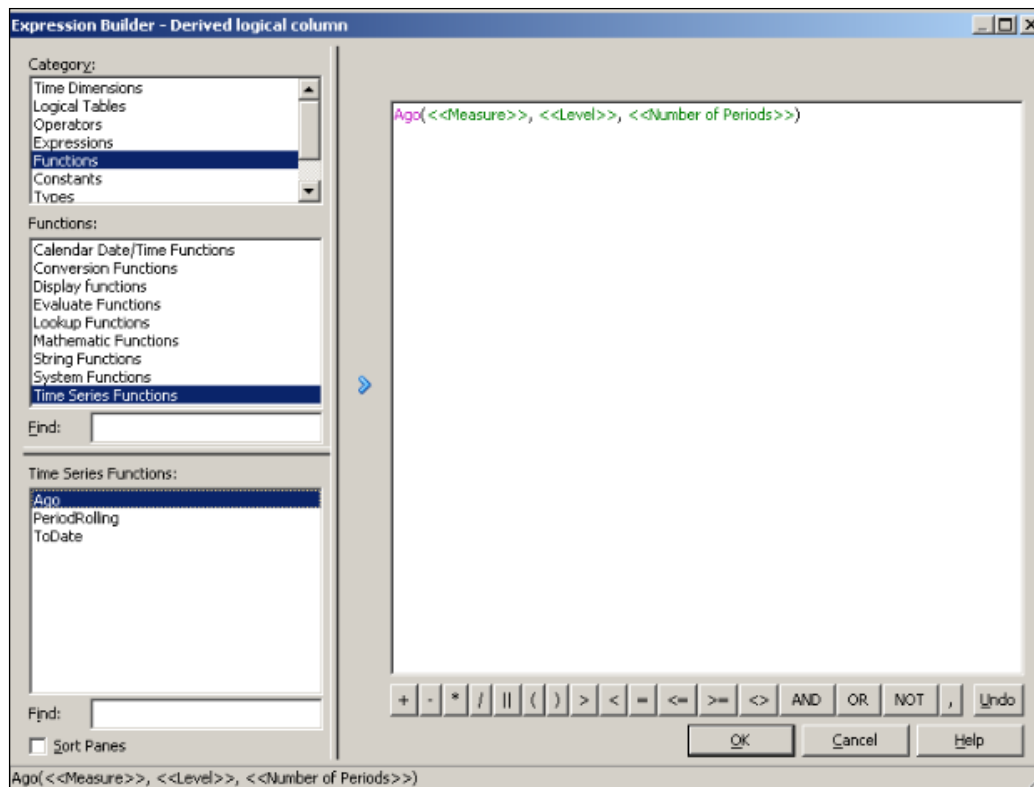
5. The input arguments will be selected as follows:

   ❑ **Measure**: `Dollars`

   ❑ **Level**: `Month`

   ❑ **Number of Periods**: `1` (for a month ago)
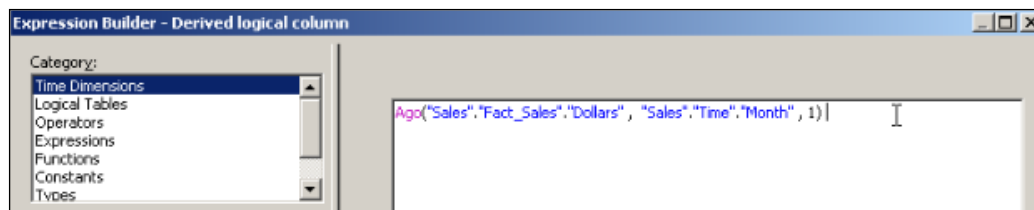
6. We're going to use similar steps to demonstrate the `ToDate` and `PeriodRolling` functions. The only different step will be on the **Expression Builder** window. So you'll find only these details on the following screenshots.

    The second function will be based on the `ToDate` function. We're going to create the `Year To Date Dollars` column.
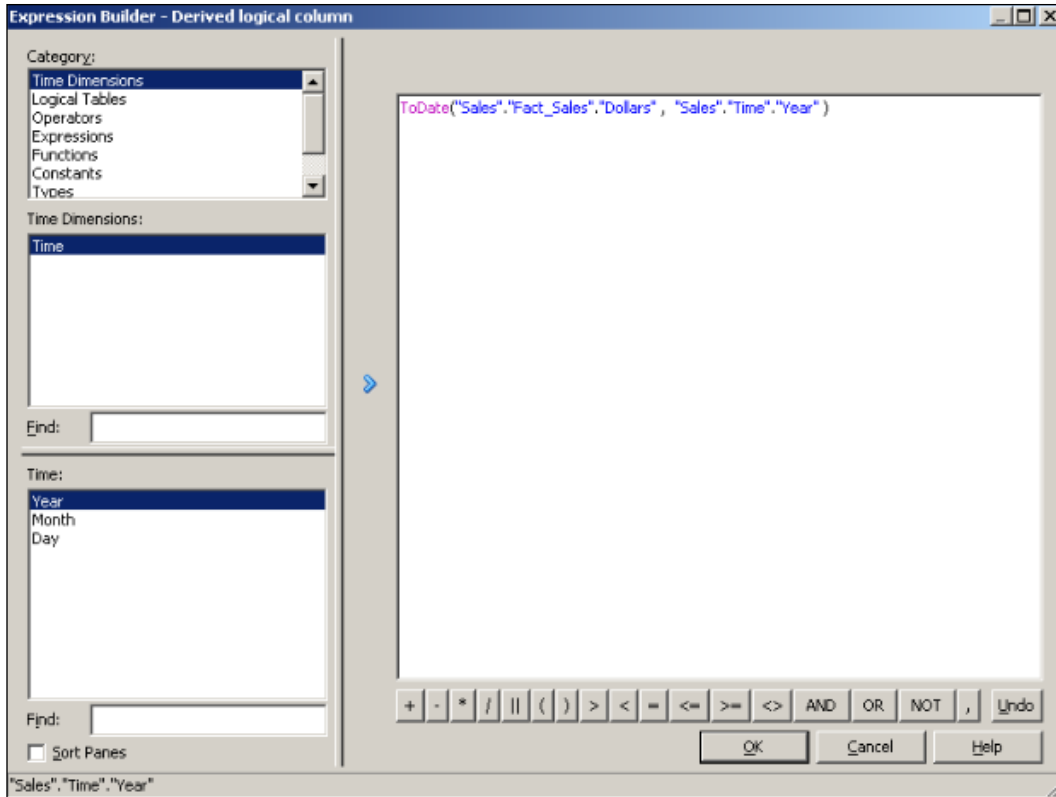
7. And the last example will be about the `PeriodRolling` function. We're going to create a new column named `3 Period Rolling Sum Dollars`.
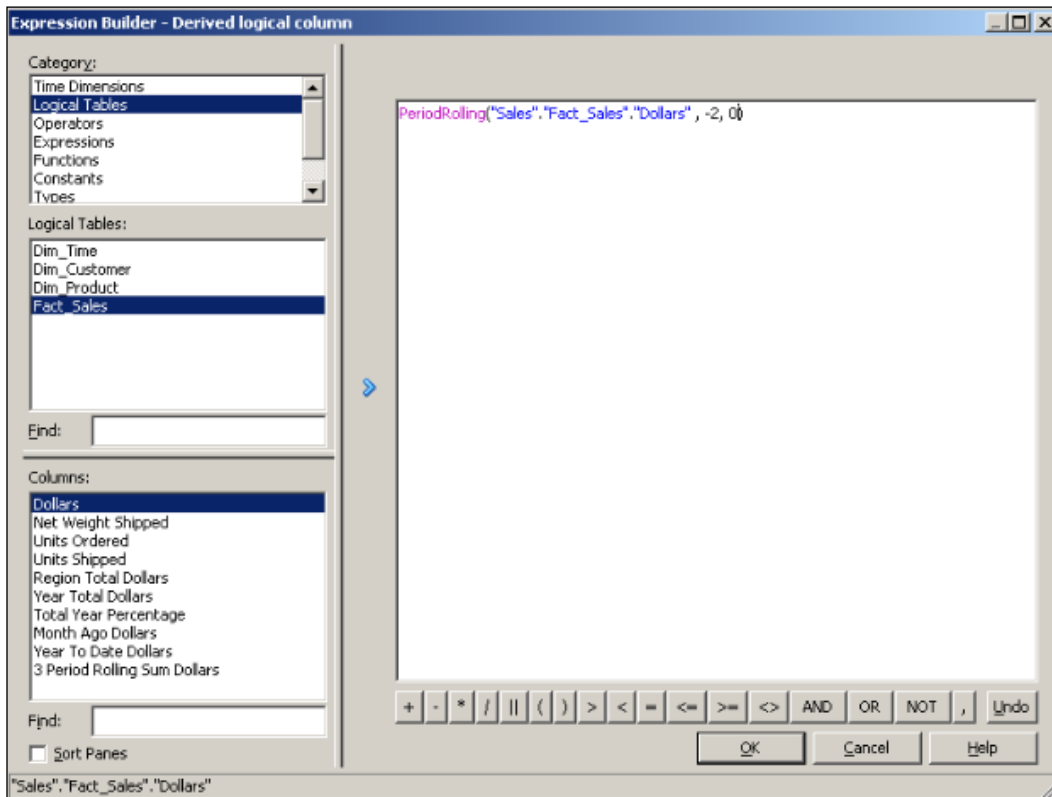
8. Obviously after creating these logical columns, presentation columns should be created by dragging them on the Presentation layer.
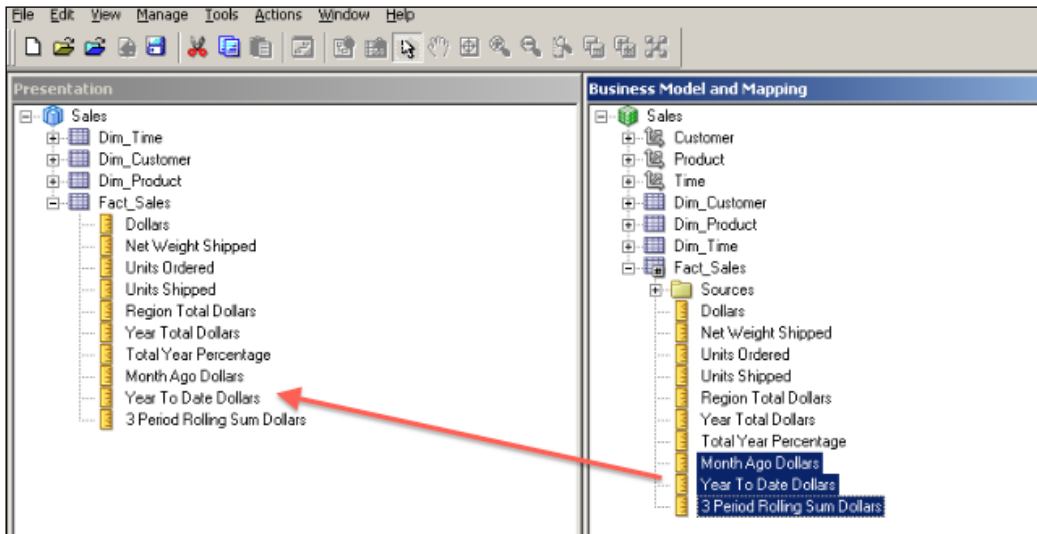
How it works...

We have created three new measure columns that are based on the time series functions. This is how these three columns look in the analysis.

MONTH	Dollars	Month Ago Dollars	Year To Date Dollars	3 Period Rolling Sum Dollars
January	3,568,665		3,568,665	3,568,665
February	3,884,407	3,568,665	7,453,072	7,453,072
March	3,975,734	3,884,407	11,428,807	11,428,807
April	3,907,255	3,975,734	15,336,061	11,767,396
May	4,061,557	3,907,255	19,397,618	11,944,545
June	3,994,531	4,061,557	23,392,149	11,963,342
July	4,054,411	3,994,531	27,446,560	12,110,499
August	4,242,611	4,054,411	31,689,171	12,291,553
September	3,810,263	4,242,611	35,499,434	12,107,286
October	4,596,372	3,810,263	40,095,806	12,649,246
November	3,655,169	4,596,372	43,750,975	12,061,804
December	3,997,616	3,655,169	47,748,591	12,249,157
Grand Total	**47,748,591**	**43,750,975**	**306,806,909**	**131,595,372**

There's more...

It's possible to calculate similar results without using time series functions, but it'll take more time and also it won't be easy to maintain them. It's recommended to use these time series functions wherever possible.

4
Working with Multidimensional Data Sources

In this chapter, we will cover:

- ▶ Importing the multidimensional data source
- ▶ Accessing members and member counts
- ▶ Creating the multidimensional Business Model
- ▶ Implementing Horizontal Federation
- ▶ Implementing Vertical Federation

Introduction

Online Transactional Processing (**OLTP**) databases are optimized for transaction processing. The performance of the `Insert`, `update`, and `delete` SQL statements are very important in the OLTP databases. They store the data in the two-dimensional structures called as tables. Enterprises need to retrieve valuable information from these databases to make strategic decisions. The databases that are optimized for queries are called as data warehouses. The need for the data warehouses is obvious for analytical reporting. They store the data in the tables like the OLTP databases. The major difference between the OLTP and data warehouse databases is the design of the tables. Normalization rules are applied in the OLTP to improve the performance of the transactions. On the other hand, data warehouses are mostly designed based on denormalization rules.

Online Analytical Processing (**OLAP**) databases store summary data and they are used to generate aggregated result sets. They store their data in multidimensional structures that are called Cubes. They store precalculated result sets so that whenever a user executes a query, the result set is generated very quickly. They consist of measures and dimensions. To use the OLAP cubes, you must first define the structure by specifying the measures and the dimension columns. After this definition, the cube should be processed. Processing the cube means loading the data from the data warehouse into the cube structure and making any possible calculations. The query language of these multidimensional sources is called **Multidimensional Expressions** (**MDX**).

Multidimensional sources are very useful although they have a cost. After implementing multidimensional sources and creating the cubes, you'll have to maintain them.

Cubes should be processed when ETL jobs are finished loading the data to core data warehouses.

Importing the multidimensional data source

Detailed data that is stored in OLTP databases should be loaded into core data warehouses and then the structure of the cubes should be created in multidimensional source databases. In our case this source will be based on Essbase Server.

Oracle Business Intelligence Enterprise Edition Server can be configured to connect to these multidimensional sources including Essbase Servers. We're not going to discuss how to build a cube in this recipe. We're going to learn how to create a repository based on the Essbase source.

How to do it...

1. We're going to use **BI Administration Tool** to import multidimensional source metadata to the Physical layer of the repository. When you click on the **Import Metadata** option from the **File** menu, the **Import Metadata** wizard will pop up. We'll have to select the **Connection Type** option as **Essbase** and the server name where Essbase is installed and configured. Additionally, **User Name** and **Password** should be entered in order to access Cubes.
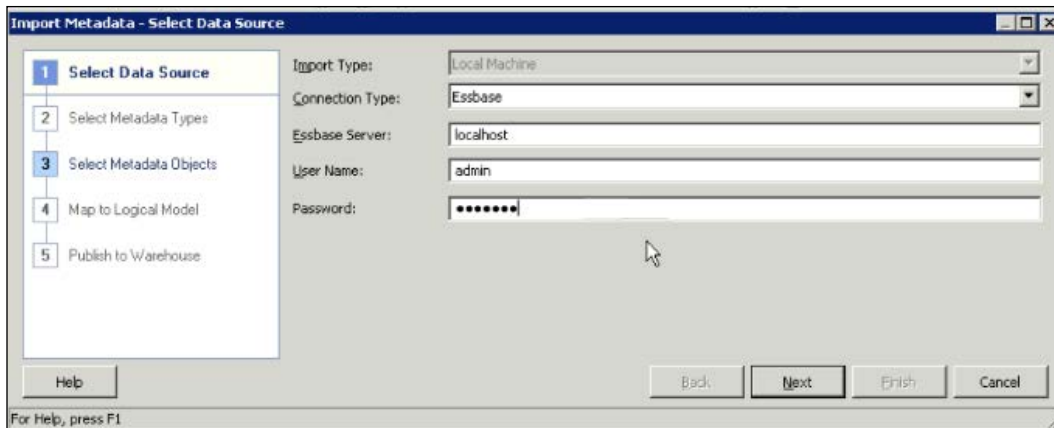
2. In the next step, you'll see the list of data sources. We're going to select the cube that we'll import to the Physical layer. In our scenario we're going to use the **Forecast** source. It contains sales related data. Also select the **Import UDAs** checkbox. **User Defined Attribute** (**UDA**) is a descriptive tag about an outline member. One UDA can be attached to many members to simplify the on-going database operations.
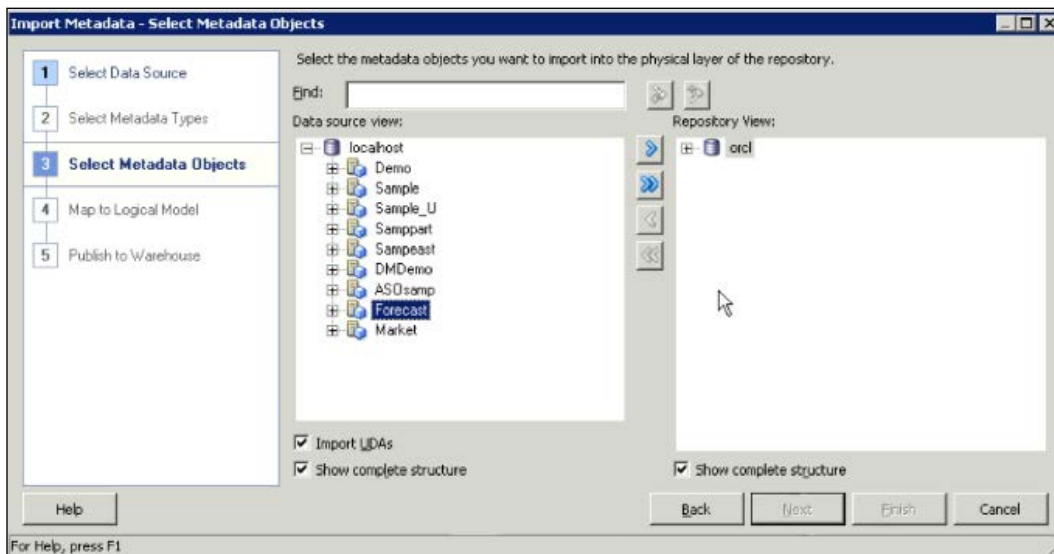
3. After selecting a cube, you'll see the dimensions and measures that will be imported.
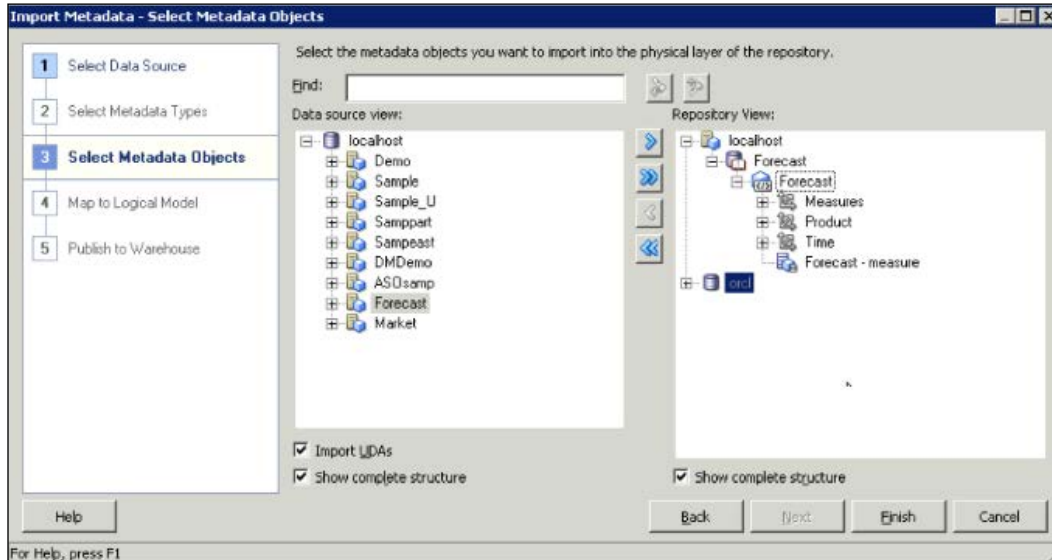
4. Clicking on **Finish** in the wizard will display the objects newly imported to the Physical layer. So **Measures**, **Product**, and **Time** dimensions are already imported. The icons of the objects are also different than table or view object's icon.
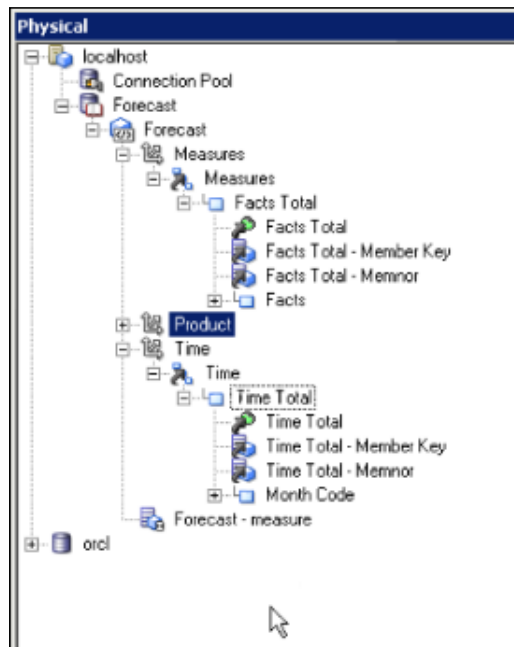
5. By default, measures are displayed like dimension levels. To change this default behavior, right-click on the cube and select **Convert measure dimension to flat measures**. So measures will be displayed in a flat level.
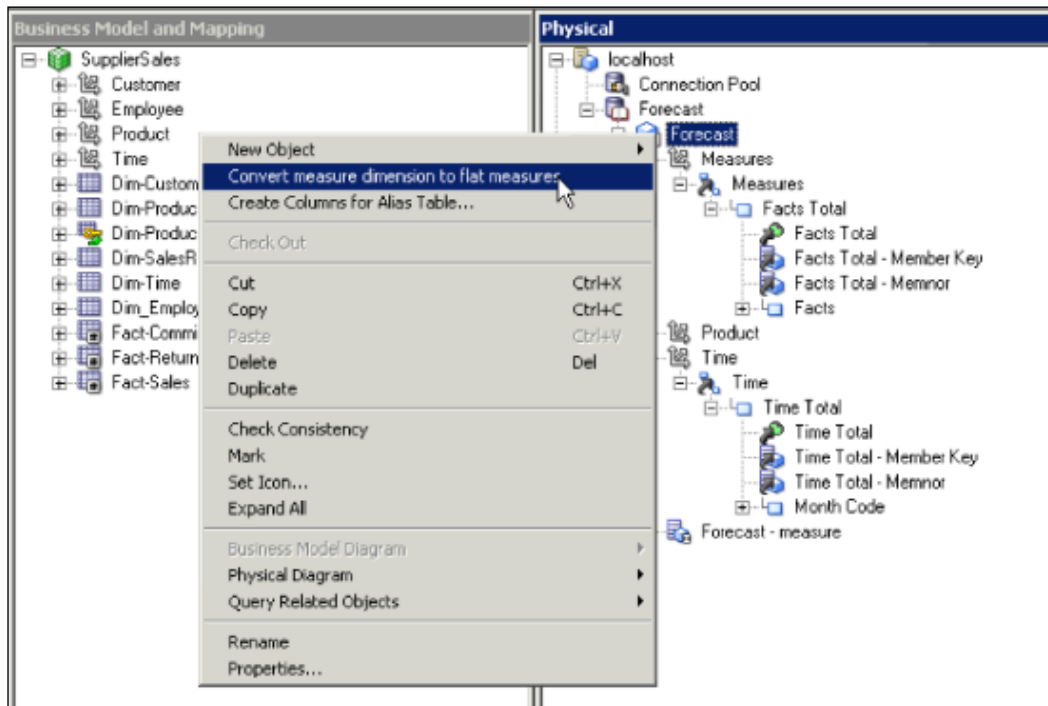
How it works...

We have just finished importing a multidimensional source. You'll notice that the structure of the physical layer looks like a Business Model in the BMM layer. But actually, it's the Physical layer.

Accessing members and member counts

It's important to understand the structure of the data in the cubes such as those in RDBMS tables. You can easily access the data from **BI Administration Tool**.

How to do it...

When you right-click on the cube, you'll see **View Members**. A new window pops up that will display the number of members. Also you're going to see the distinct values. This result set is accessed from the multidimensional source.
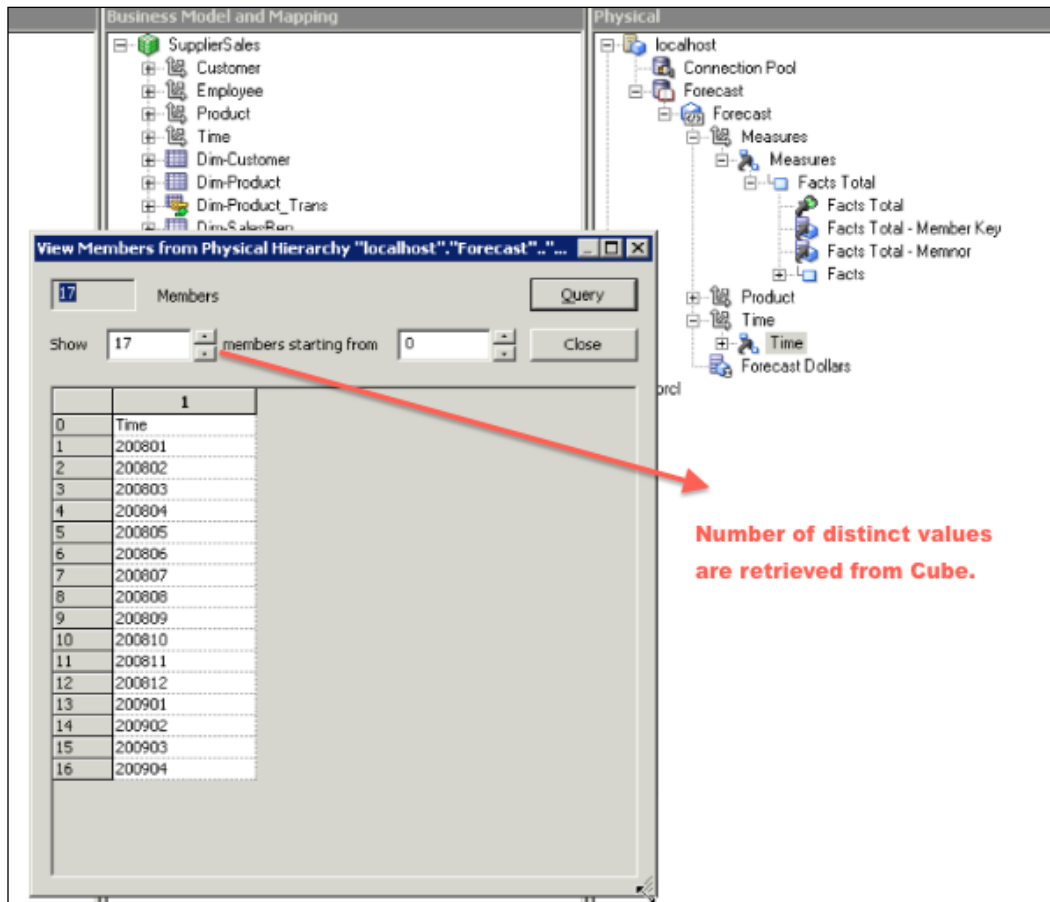
How it works...

BI Server generates an **MDX query** so that it displays the number of attribute values. MDX and SQL languages are used to query databases. SQL statements are used to construct a relational view. MDX generates multidimensional views that consist of multiple-dimension attributes in the result set. While the SQL statements use tables as sources, MDX uses Cubes. The structure of the MDX statement is composed of the following clauses:

▸ `WITH` (optional): It allows calculated members to be computed during the processing

- ▸ SELECT: It identifies the dimension members on each axis
- ▸ FROM: The names of the cubes that are being queried
- ▸ WHERE (optional): It defines which dimension is used as a slicer

Creating the multidimensional Business Model

Creating a Business Model for cubes is easier than RDBMS tables. Actually, the structure of cubes is very similar to the Business Models. Cubes contain dimensions and measures. We can see that structure in the Physical layer as well.

Dimensions and their levels are already created in the cubes. Measures are also specified in the cubes. So all required object definitions already exist in the cubes.

How to do it...

1. First step will be creating a blank Business Model in the BMM layer. Then we're just going to drag-and-drop a cube from the Physical layer onto the new Business Model. At this step, all logical dimensions and fact tables will be created automatically, plus dimensions and their hierarchies are also created in only one step.
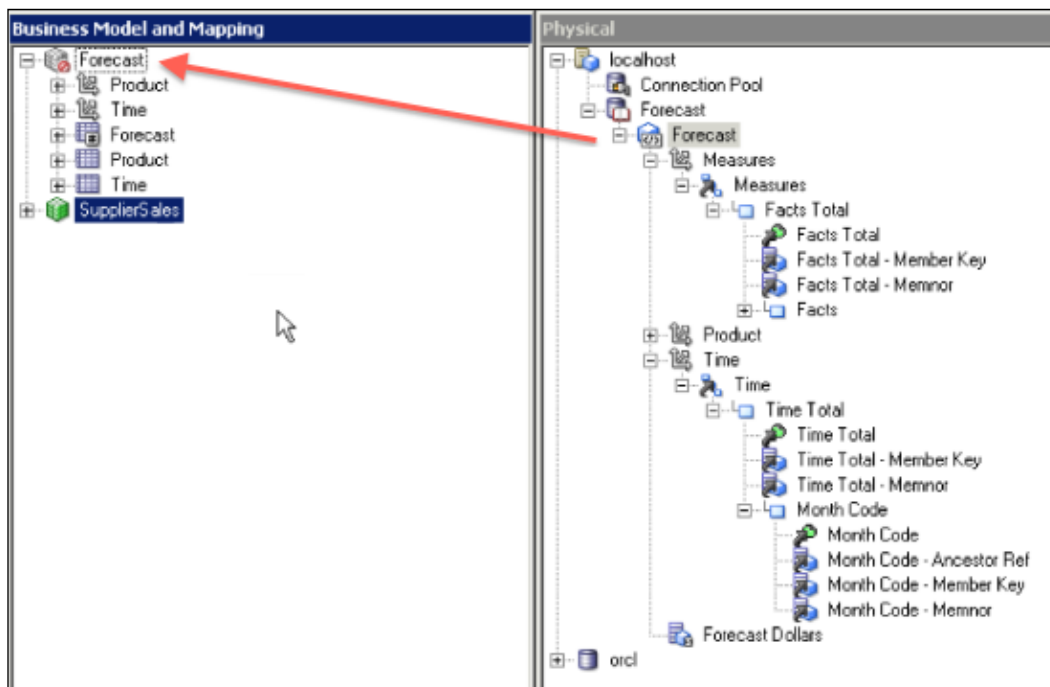
2. Now we're going to clean up the **Forecast** Business Model. Unneeded columns should be removed from the model. **Product Type - Memnor** and **Product Type - Member Key** columns will be removed in the following screenshot:
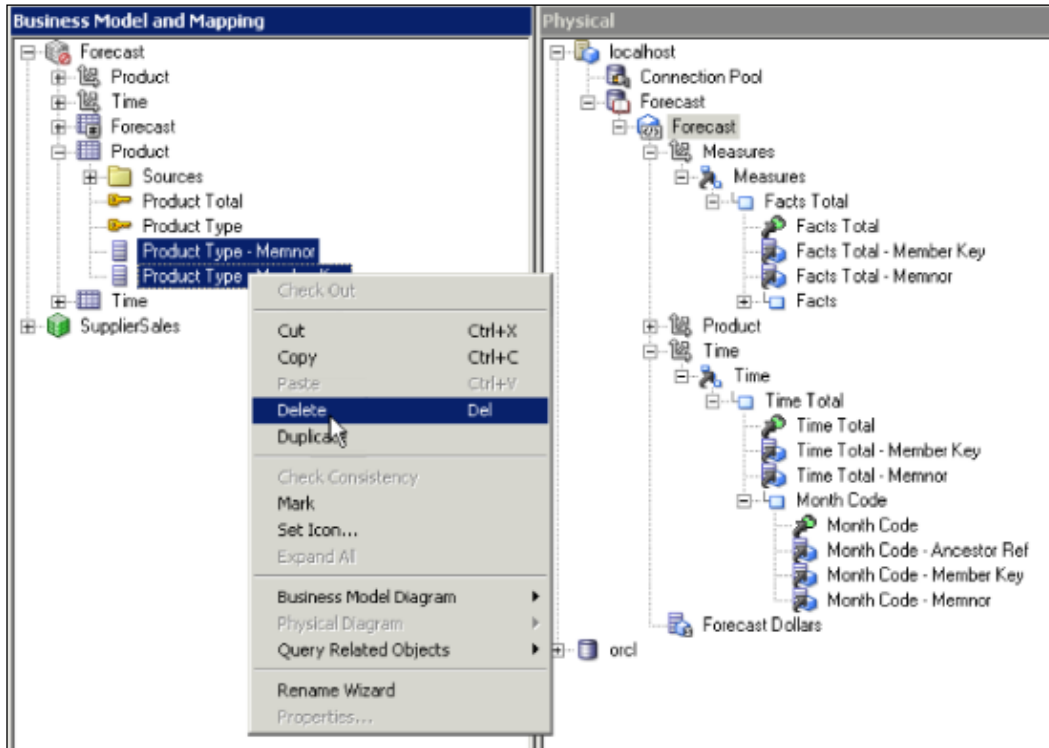
3. We're going to remove member key and memnor columns from the **Time** logical dimension table. Normally these member keys (and no columns) are required in cubes and not in Business Models.
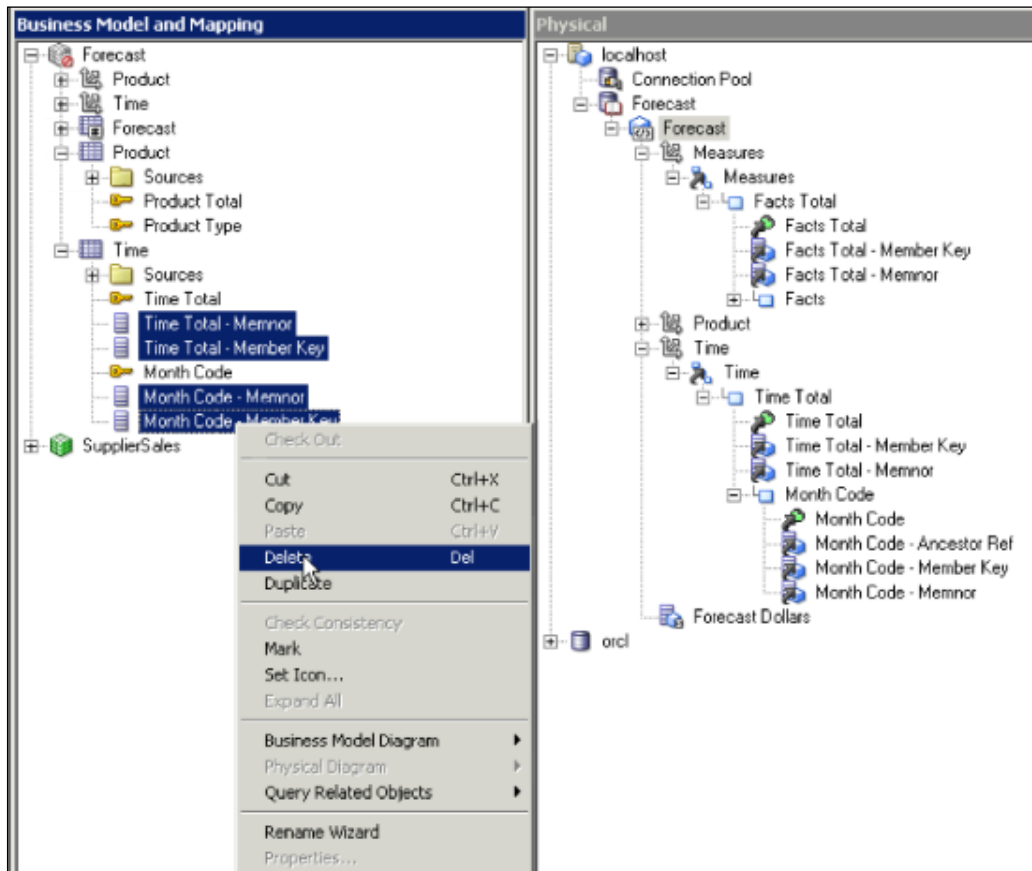
4. As a result, we have finished creation of a new Business Model. Two dimensions with their hierarchies are created. The Business Model includes three logical tables:

   ❑ **Forecast**: Logical fact table

   ❑ **Product**: Logical dimension table

   ❑ **Time**: Logical dimension table
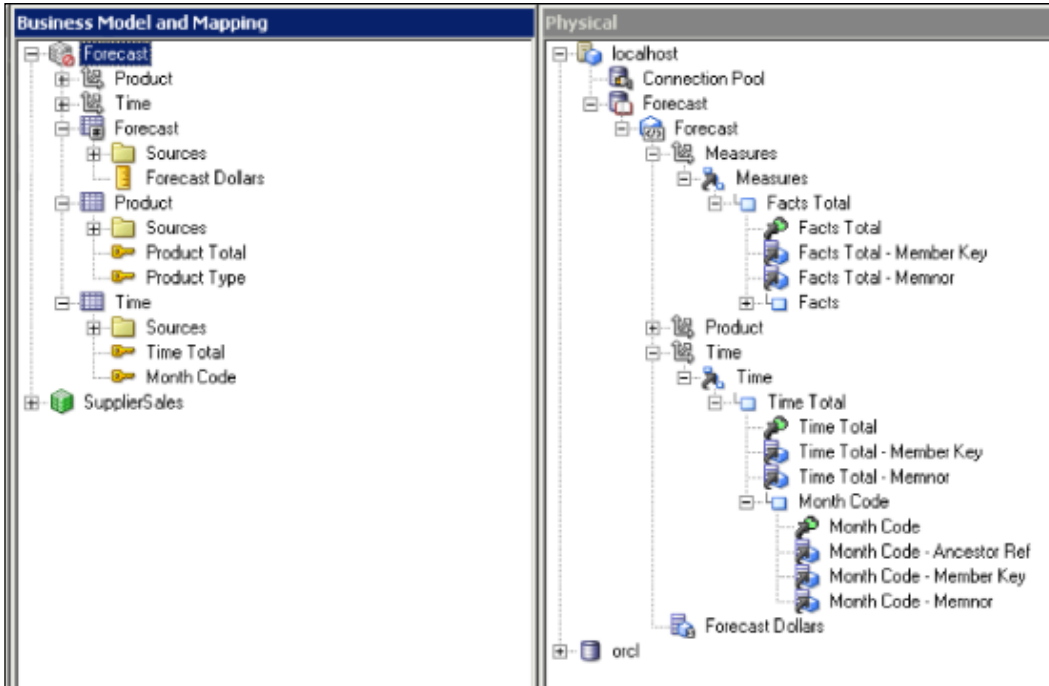
5. But the new Business Model is still inconsistent, although everything is correct in the model. The only missing definition in the repository is **Subject Area**. **Forecast Business Model** is not mapped with any **Subject Area**. So this should be created in the Presentation layer. This is another easy step. Just drag-and-drop the Business Model into the Presentation layer.
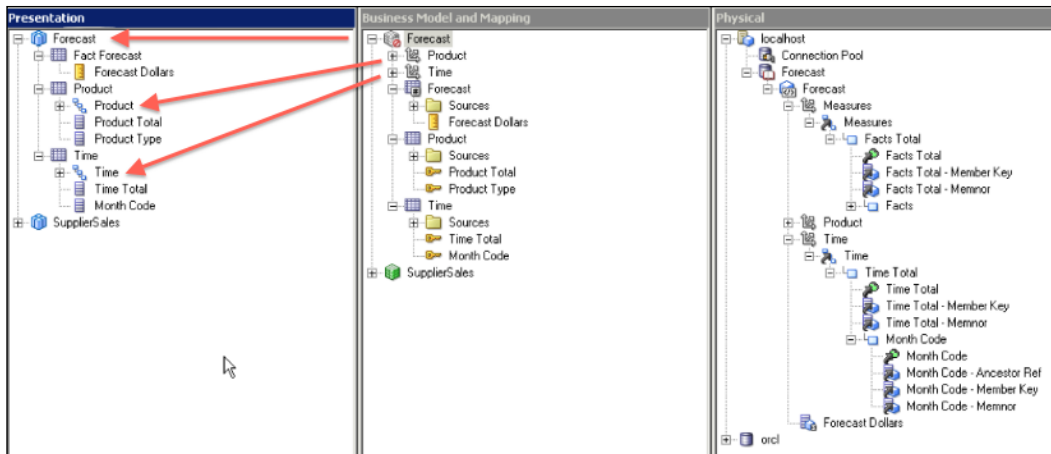
6. Presentation tables, columns, and hierarchies will be automatically created in the **Presentation** layer. And at last, the **Forecast** Business Model will be consistent so it will be available for queries.
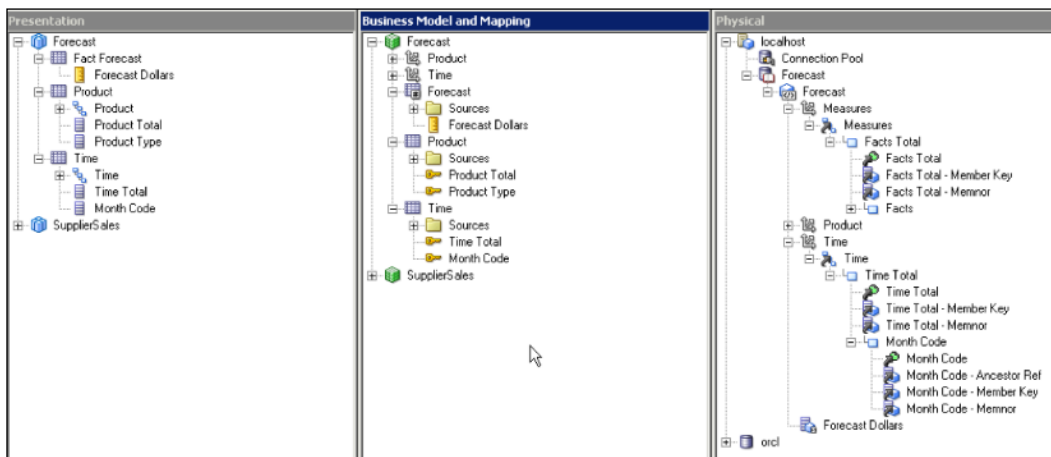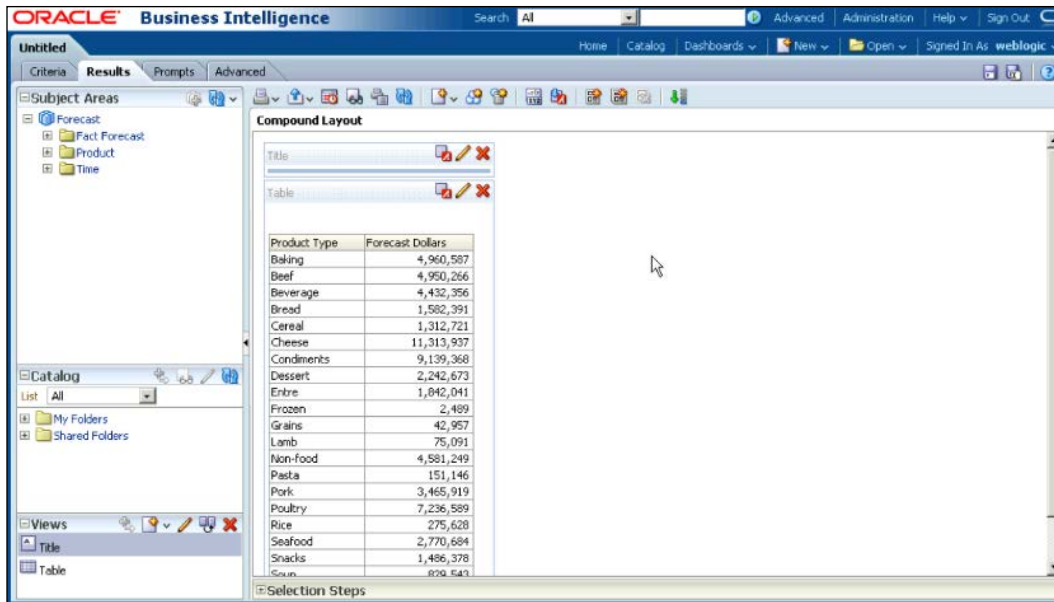
How it works...

Now it's time to test the Business Model. We're going to verify if the new repository is ready and available in Presentation Services. When you log into BI Presentation Service and start to create a new analysis, you'll see that the **Forecast** subject area is available to end users.
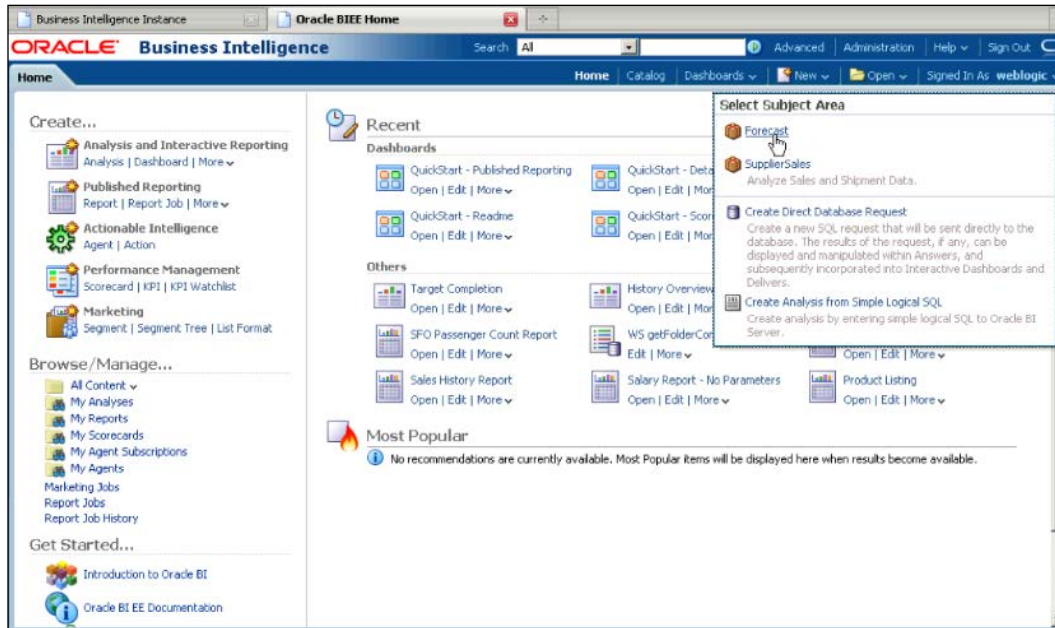
As a simple test, we're going to add **Product Type** and **Forecast Dollars** to the new analysis and check the result set. As you'll see in the following screenshot, the result set is retrieved from the **Forecast** cube that is a multidimensional source. Also when you check the query log, you'll see that MDX query is generated by the BI Server and executed against the Essbase Server.

There's more...

Multidimensional source is integrated into the BI Repository and it's working perfectly. Now we can gain benefit of some reports that will show us Forecast and Actual values. As you already noticed, the Forecast cube stores projected values. On the other hand, actual values are stored in the RDBMS tables. We're going to learn how to integrate cube and RDBMS sources in the Business Model in the next recipe.

Implementing Horizontal Federation

As we discussed in the beginning of this chapter, OLAP cubes store precomputed summary data. Business users may need to compare the values that are stored in the cubes with the values from the relational databases. Combining the result sets from two different data sources can solve this business challenge.

Horizontal Federation allows us to generate a result set that is retrieved from both multidimensional and relational data sources. We'll have to make some modifications in the Business Model in order to implement Horizontal Federation.

In our scenario, projected values will be retrieved from multidimensional sources. The **Forecast Dollars** column will be the example. And we're going to use the **Dollars** column from the relational data source. So the business users will be able to compare the actual and projected values in the same report.

How to do it...

1. Let's check the logical table sources of the **Forecast** table. You'll see that there's only one source and it's already mapped to multidimensional source.
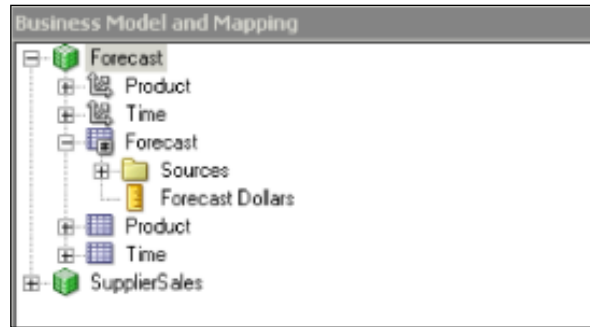
2. Drag-and-drop the **Dollars** physical column from the physical layer onto the **Forecast** logical table.
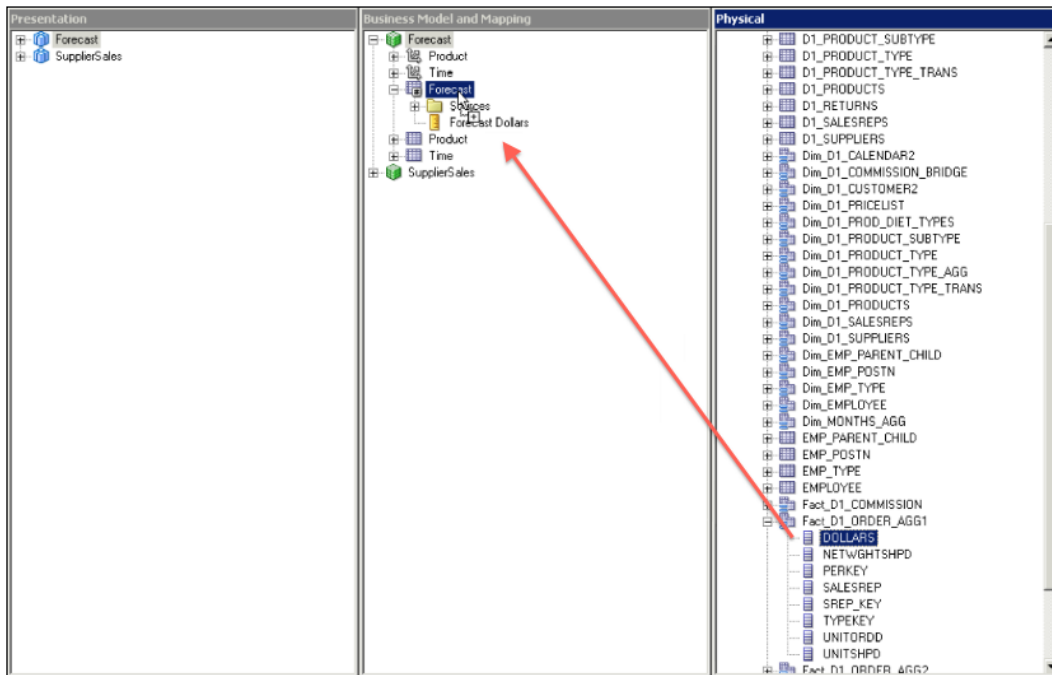
3. The last action is going to create the second logical table source in the **Forecast** logical table. The new source is automatically mapped with the relational table.
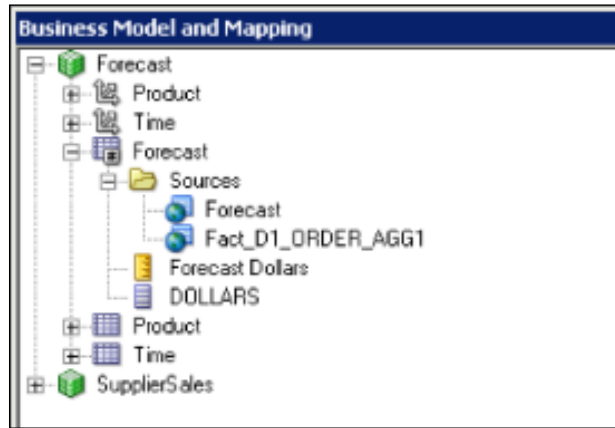
4. Repeat the step for the remaining dimension logical tables. Second logical table sources should be created for both **Product** and **Time** tables. First we're going to make the configuration in the **Product** table.
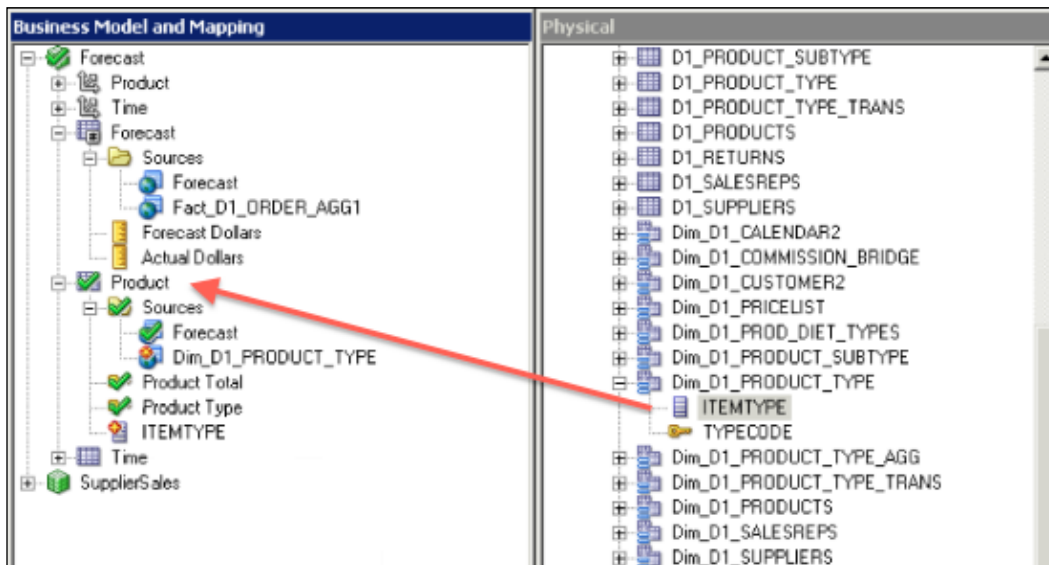
5. Then the same step will be repeated for the **Time** table as shown in the following screenshot. Now all the logical tables have two logical table sources.
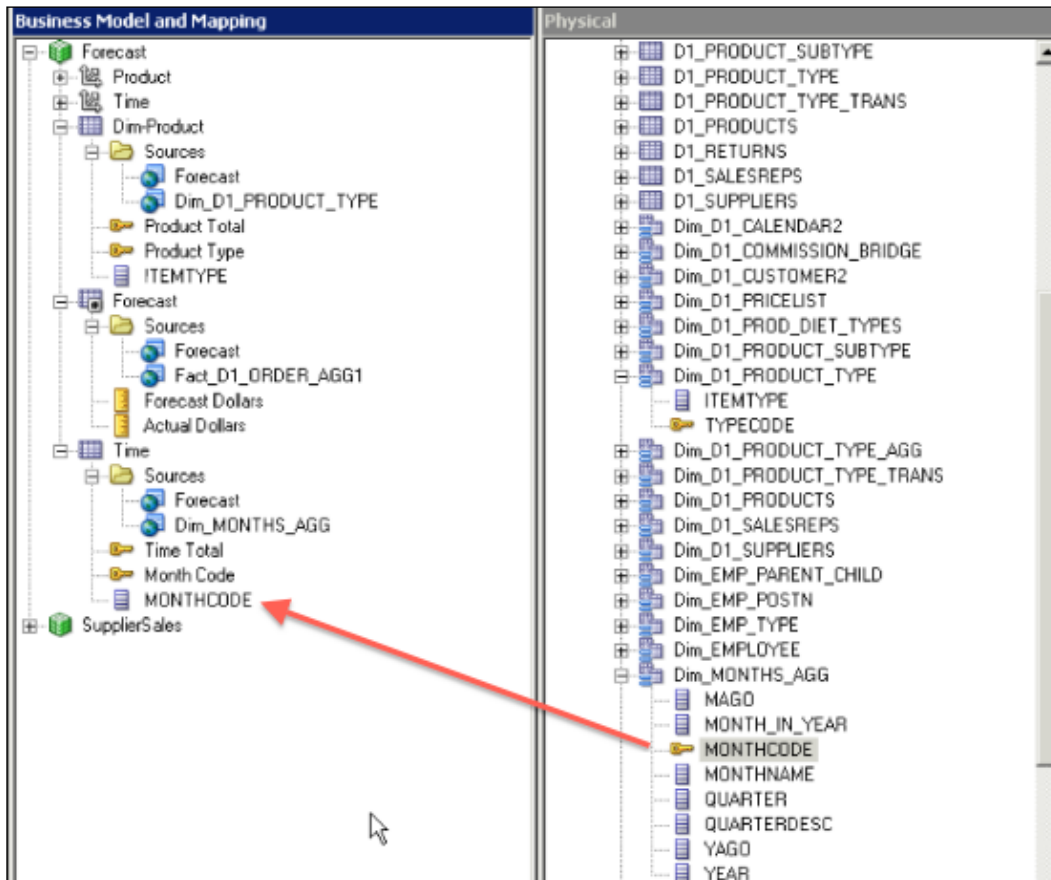
6. Obviously, in order to implement Horizontal Federation correctly, we'll have to create required presentation columns in the Presentation layer. In our example, the **Actual Dollars** column is going to be created and mapped to the **Actual Dollars** logical column in the Business Model and Mapping layer.
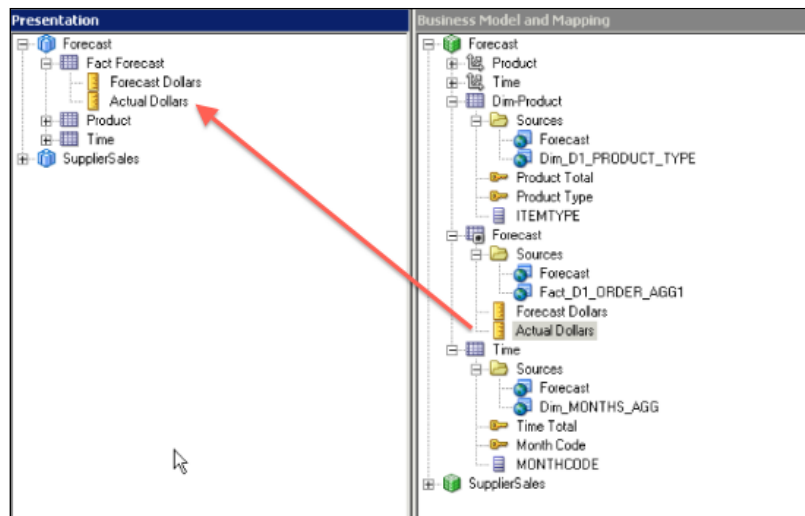
How it works...

In order to test the new repository, we've added **Product Type**, **Forecast Dollars**, and **Actual Dollars** from the selection pane to the analysis. You'll see that the BI Server generates the result set correctly. It accesses both sources to generate the result set. The **Forecast Dollars** column values are populated from multidimensional source and the **Actual Dollars** column values are from the relational database.

There's more...

You can also check the query log file. You'll find two queries were executed against two different data sources.

Implementing Vertical Federation

Multidimensional sources store aggregated data, not the detailed data. Because of their nature, they just contain summary data at certain levels. These levels are specified during creation of the cubes.

Vertical Federation provides the ability to drill through aggregate multidimensional data into detail relational data.

We're going to use a different cube named `Market` to demonstrate Vertical Federation. As you've already learned how to import the multidimensional metadata and how to create a business model, these steps will be skipped in our example.

Here's the new source and new Business Model:

The new cube contains Dollars values that are aggregated at these levels:

- ▶ **Area dimension**: **District** level
- ▶ **Time dimension**: **Year** level

Relational source contains **Dollars** values that are aggregated at these levels:

- ▶ **Area dimension**: Sales person level (lower detail level than the **District** level)
- ▶ **Time dimension**: Year level
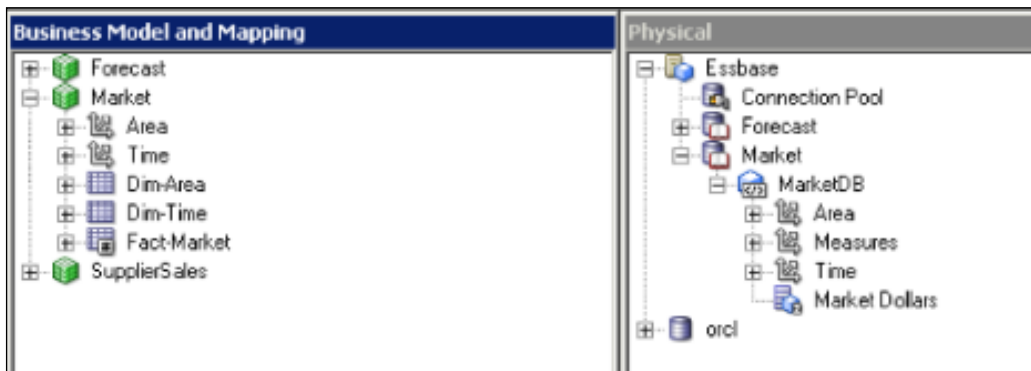
How to do it...

1. We're going to create a second logical table source for the fact logical table. The first logical table source is mapped with a multidimensional source. The second one is mapped with a relational data source.
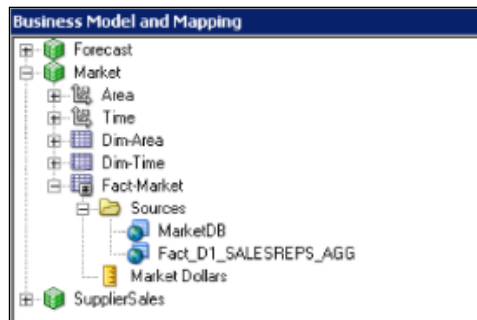
2. As you'll remember, whenever a logical table has more than one logical table source, you should give instructions to the BI server about the usage of these sources, so we're going to specify the same level that the aggregation is done in the source. You'll see both levels are set for the multidimensional source properties. The levels are specified in the **Content** tab of the logical table source.
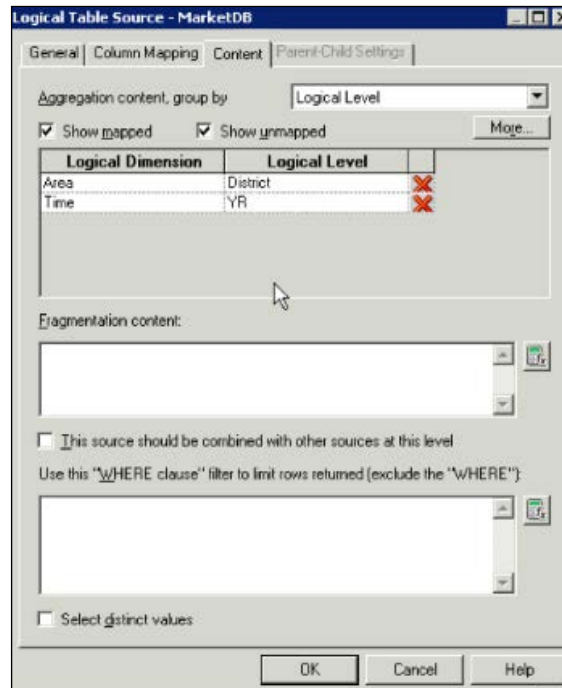
3. Similar action is going to be repeated for the second logical table source that is mapped with a relational data source. But this time the **Sales Rep** level is going to be selected in the **Content** tab in the **Logical Table Source – Fact_D1_SALESREPS_ AGG** window because the relational data source contains aggregated values at the **Sales Person** level.
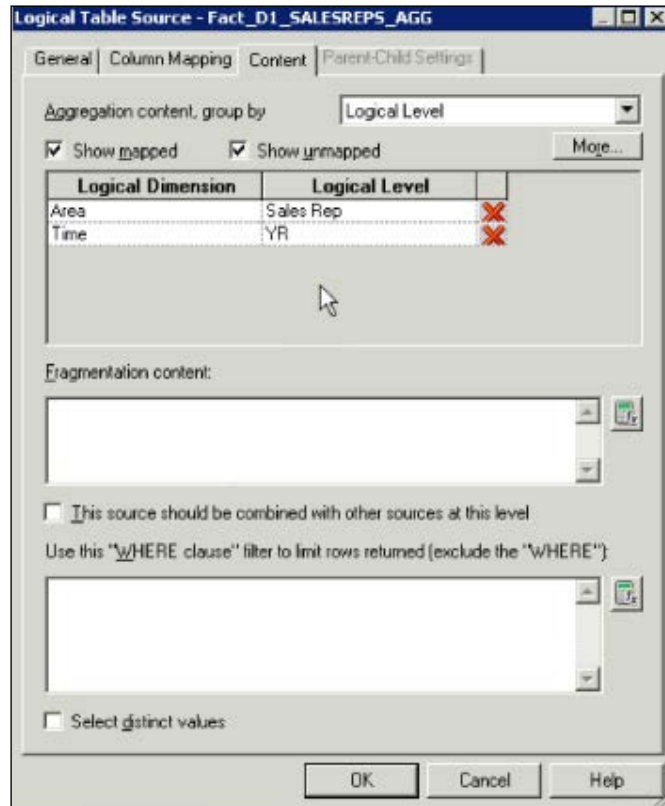
4. We'll have to repeat these steps for the remaining logical tables. Second logical table sources should be defined that will be mapped to relational data sources and also contents should be defined. In addition to these settings, the **Sales Rep** logical level should be created in the hierarchy because by default the cube doesn't contain any data below the **District** level. Also, required new presentation columns are going to be created in the Presentation layer.

```
Business Model and Mapping
⊞ Forecast
⊟ Market
   ⊟ Area
      ⊟ Area Total
         └ Area Total
         ⊟ Region
            └ Region
            ⊟ District
               └ District
               ⊟ Sales Rep
                  └ Sales Rep
   ⊞ Time
   ⊟ Dim-Area
      ⊟ Sources
         MarketDB
         Fact_D1_SALESREPS_AGG
      Area Total
      Region
      District
      Sales Rep
   ⊞ Dim-Time
   ⊞ Fact-Market
⊞ SupplierSales
```
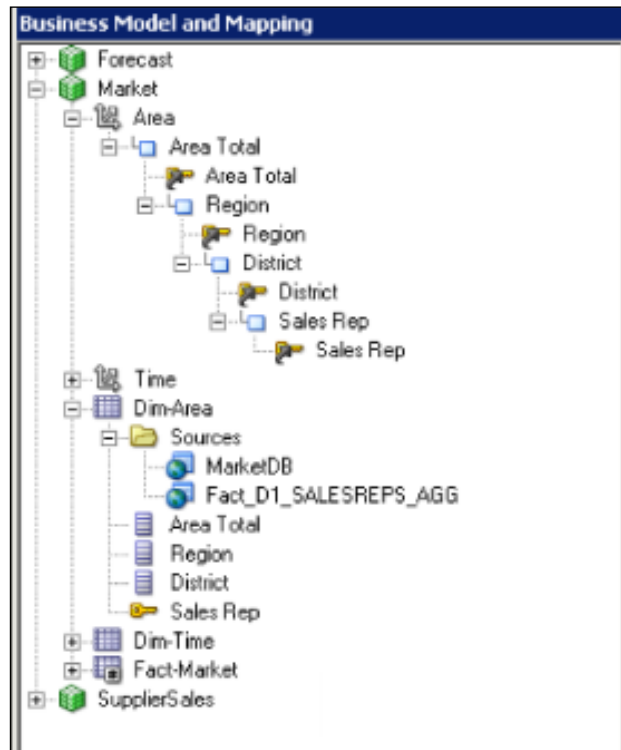
How it works...

Now we're going to test the new modifications. We're going to create a simple analysis that will contain two attribute columns and one measure column.

▸ **Region**: Mapped to the multidimensional source

▸ **District**: Mapped to the multidimensional source

▸ **Dollars**: Mapped to the multidimensional source

You'll see that the result is generated from the multidimensional source.
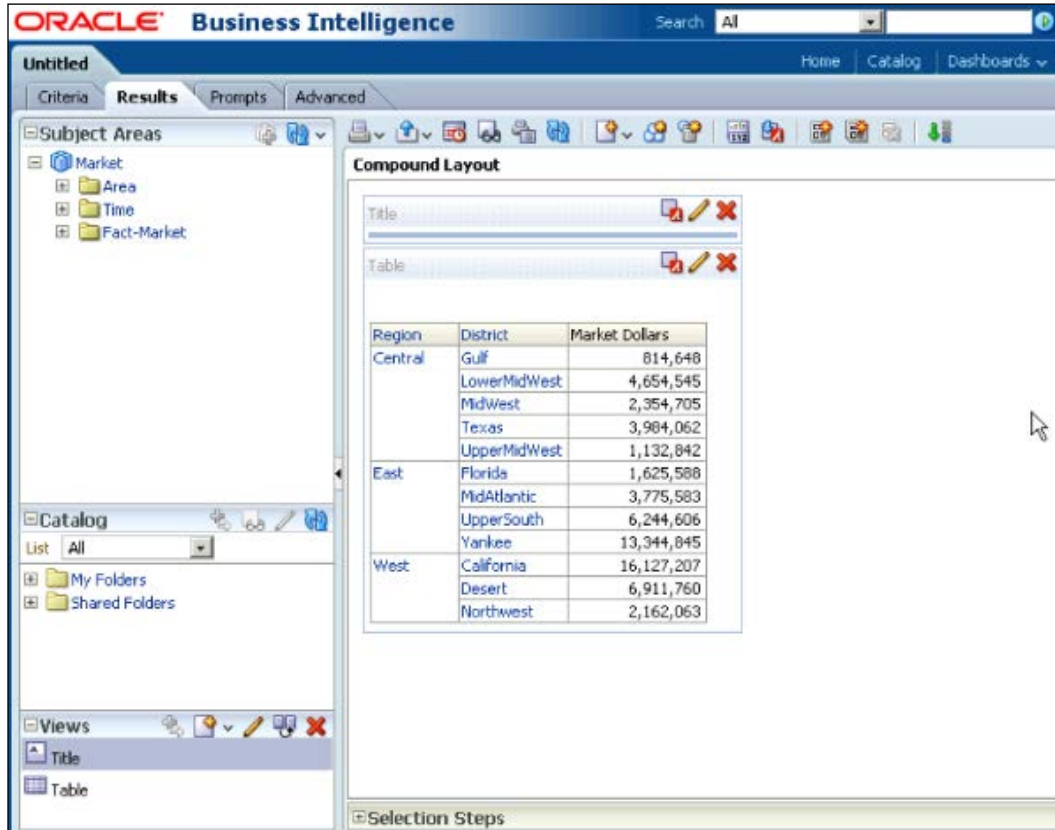
But if the end user wants to drill down through the **District** level, the multidimensional source won't be able to satisfy the entire query. The **Sales Person** hierarchy level is mapped to the relational column.

So this time you'll gain the benefit of Vertical Federation. Lower detail column (**Sales Rep**) values will be retrieved from relational data sources and the rest will be retrieved from the Essbase cube.

There's more...

Designing the structure of the cubes in Data Federation is quite important. You won't be able to store all the details in the cubes. The duration of the cube process can take long hours so we have to decide about the data granularity that will be stored in the cubes. We need to monitor the user behaviors. Enabling usage tracking in the repository allows us to monitor the statistics about report executions. This will be covered in *Chapter 6, Managing Usage Tracking and Enabling the Cache.*

5
Security in Oracle BI

In this chapter, we will cover:

- ▶ Configuring security settings
- ▶ Creating users
- ▶ Creating groups
- ▶ Creating application roles
- ▶ Setting up permissions on repository objects
- ▶ Configuring query limits
- ▶ Specifying the time restrictions
- ▶ Creating data filters

Introduction

In Business Intelligence projects, configuring security settings is one of the most important steps. Business users should be authenticated and they should only access the data which they need. In order to do this, we have to configure the authentication and the authorization methods.

The default authenticator is the WebLogic Server. We already discussed that OBIEE 11*g* is integrated with the WebLogic Server. The user accounts are going to be created in the WebLogic Server. Optionally, LDAP authentication can be configured. The advantage of LDAP authentication is that we don't need to create user accounts in WebLogic. Instead of that, we can use the existing user accounts that are already created on the LDAP servers. In our sample scenario, we are going to use the default authentication so that we can create the user accounts. In order to manage the security, we're going to use groups. User accounts and groups will be created in the **WebLogic Server Administration Console** window.

Then we need to create the Application Roles and map the application roles with the groups. Application roles will be created and managed in **Oracle Enterprise Manager Fusion Middleware Control**. Enterprise Manager is used to manage the applications that are integrated with WebLogic Server. OBIEE 11*g* is one of the applications that can be managed by Enterprise Manager.

After the authentication configuration, it will come to authorization rules. We are going to grant privileges to the user accounts or to the application roles in the BI server repository. Obviously, we're going to use **BI Administration Tool** to achieve this.

As a summary you can find the security mapping below:

USER ACCOUNTS → GROUPS ←→ APPLICATION ROLES ← PERMISSIONS

Security permissions can be implemented in three ways:

- ▶ **Object-level security**: We can set permissions on objects such as subject areas, presentation tables, columns, and so on.

- ▶ **Data-level security**: We can define data filters to eliminate some of the rows from the result set. Let's assume that there are two region managers and they both need to access the fact table, but they should see only the region data that they are responsible for.

- ▶ **Presentation Catalog security**: We can set permissions on the Presentation Catalog objects such as dashboards, analyses, KPIs, and so on.

You're going to find all the details about security in this chapter.

Configuring security settings

We're going to create user accounts and groups in **WebLogic Administration Console** and create the application roles in **Oracle Enterprise Manager Fusion Middleware Control**. Then we're going to import the user accounts and the application roles into the repository by using **BI Administration Tool**.

We're going to learn how to access the **Administration Console** window and the **Enterprise Manager** window.

How to do it...

1. Login to **Oracle WebLogic Server Administration Console** in order to create users and groups. The default port number of the **Administration Console** window is 7001. The web address is `http://servername:7001/console.`

2. Once you log in, click on the **Security Realms** link. You will find only one realm by default, called **My Realm**. You'll see the list of the user accounts and the groups. Then create the user accounts and the groups and configure the membership of the groups. We're going to use these groups to map them with the application roles.
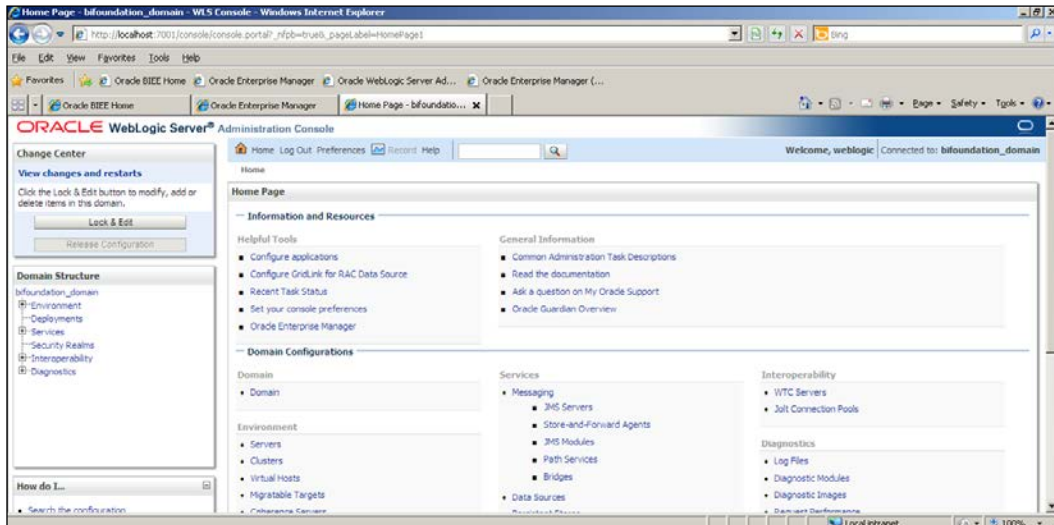
3. Login to **Oracle Enterprise Manager Fusion Middleware Control** to create application roles by accessing the web page at `http://servername:7001/em`.
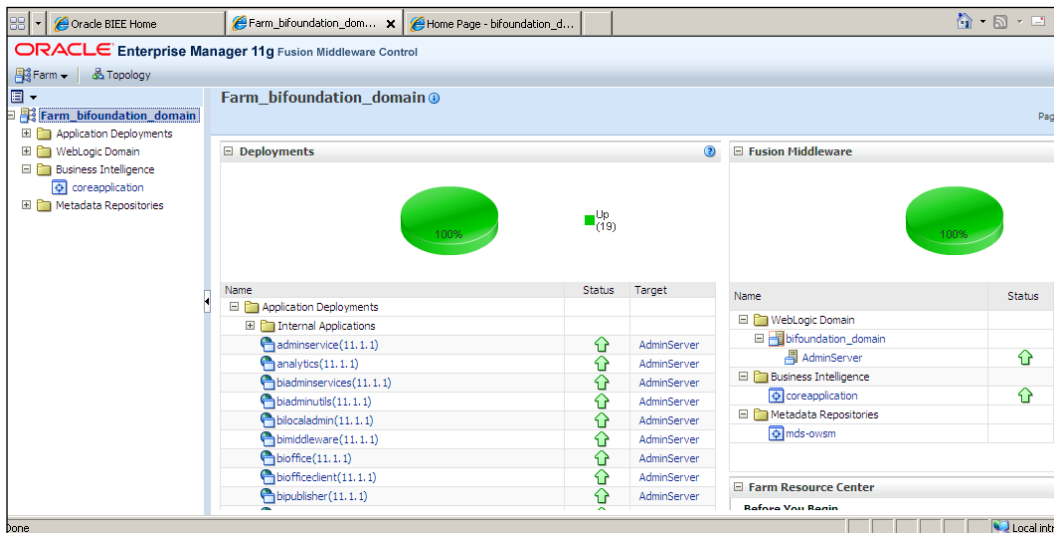
4. Open the Oracle **BI Administration Tool** to set up permissions on repository objects. Then open **Identity Manager** from the **Manage** menu.
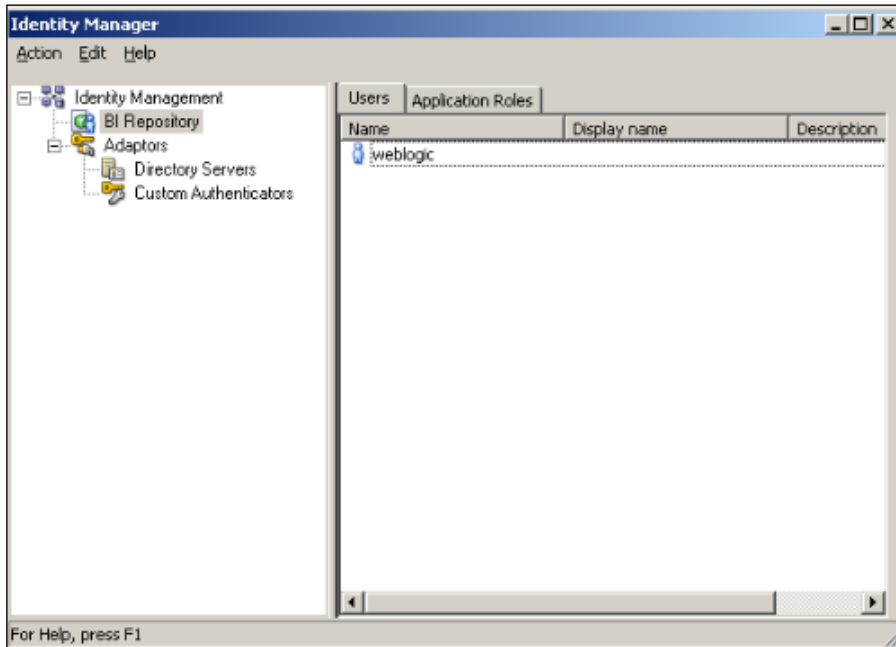
How it works...

As the first step, we have to create user accounts and groups by using **WebLogic Server Administration Console**. User accounts are going to be members of groups. Later on, they will come to application security. We're going to create application roles and map these roles to the groups.

After all these steps, user accounts and application roles are accessible from **Identity Manager** and we're going to set up permissions in the repository.

There's more...

We should also think about the security settings in the Presentation Catalog. We can either use the Presentation Services web interface or **Catalog Manager** to set up permissions on the catalog objects.

Creating users

Business users should be authenticated when they need to access Presentation Services. Oracle BI is configured to use the directory server embedded in **Oracle WebLogic Server**. WebLogic Authentication Provider is used by default. We're going to create users by accessing **Oracle WebLogic Server Administration Console**.

How to do it...

1. Once you are logged into **Oracle WebLogic Server Administration Console**, you'll see the **Domain Structure** pane. Click on the **Security Realms** link to access the list of realms in the domain.

2. Click on `myrealm` to access all the security settings. You'll also see the **Users and Groups** tab.
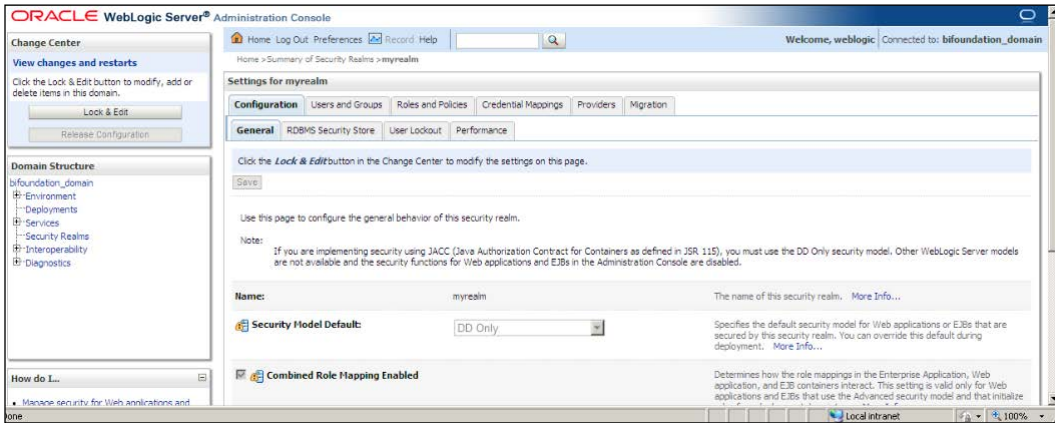
3. Go to the **Users and Groups** tab and then click on the **Lock & Edit** button. We cannot modify security settings without locking. You'll also have to click on **Release Configuration** after finishing modifications.
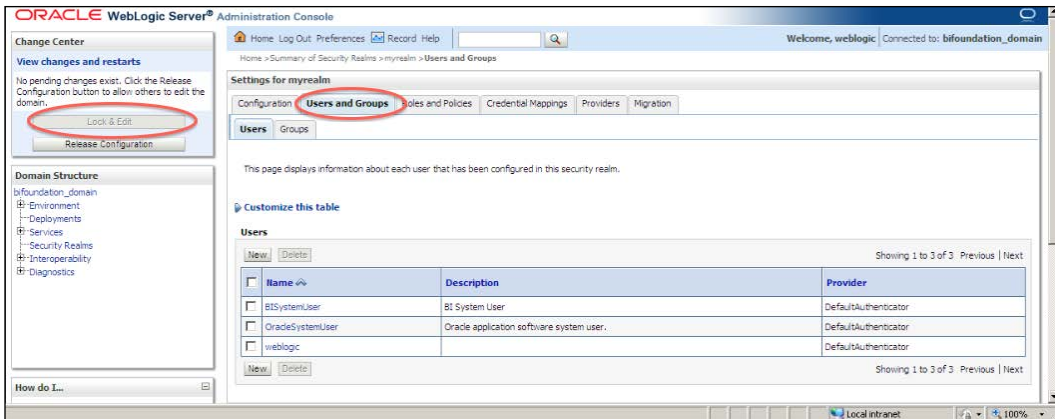
4. We're going to enter the username and password in the new window. Once it's finished, click on **OK**.

How it works...

Now that the user is created, it's time to create groups and make the user a member of the new group.

There's more...

Instead of assigning privileges to user accounts, we should assign it to groups to manage security settings in an efficient and secure way.

Creating groups

Again we're going to use the **WebLogic Administration Tool** to configure the group settings. Groups are going to be created in the **Users and Groups** tab, as well. We're going to create a group named `Sales Managers` in our sample scenario. The groups can used in any application that is integrated with the WebLogic Server.

How to do it...

1. Click on the **Groups** tab in the **Settings for myrealm** page to access the list of existing groups.

Settings for myrealm

| Configuration | **Users and Groups** | Roles and Policies | Credential Mappings | Providers | Migration |

| Users | **Groups** |

This page displays information about each group that has been configured in this security realm.

▷ **Customize this table**

Groups

| New | Delete | | Showing 1 to 10 of 11 Previous | Next |

□	**Name** △	**Description**	**Provider**
□	AdminChannelUsers	AdminChannelUsers can access the admin channel.	DefaultAuthenticator
□	Administrators	Administrators can view and modify all resource attributes and start and stop servers.	DefaultAuthenticator
□	AppTesters	AppTesters group.	DefaultAuthenticator
□	BIAdministrators	BI Administrators Group	DefaultAuthenticator
□	BIAuthors	BI Authors Group	DefaultAuthenticator
□	BIConsumers	BI Consumers Group	DefaultAuthenticator
□	CrossDomainConnectors	CrossDomainConnectors can make inter-domain calls from foreign domains.	DefaultAuthenticator
□	Deployers	Deployers can view all resource attributes and deploy applications.	DefaultAuthenticator
□	Monitors	Monitors can view and modify all resource attributes and perform operations not restricted by roles.	DefaultAuthenticator
□	Operators	Operators can view and modify all resource attributes and perform server lifecycle operations.	DefaultAuthenticator

| New | Delete | | Showing 1 to 10 of 11 Previous | Next |

2. Click on the **New** button and enter the details:

   ❑ **Name**: `SalesManagers`

   ❑ **Description**: `Test Group`

3. Once you have finished, click on **OK**.

4. The next step is to add the user into the group, in this case `SalesManagers`. There are also default groups that are created with the installation of OBIEE. We're going to add the user into the default groups too. Click the **Users** list and edit the new user by clicking on its name. Then click on the **Groups** tab and select the groups to make the membership. Click on the **Save** button after you finish.
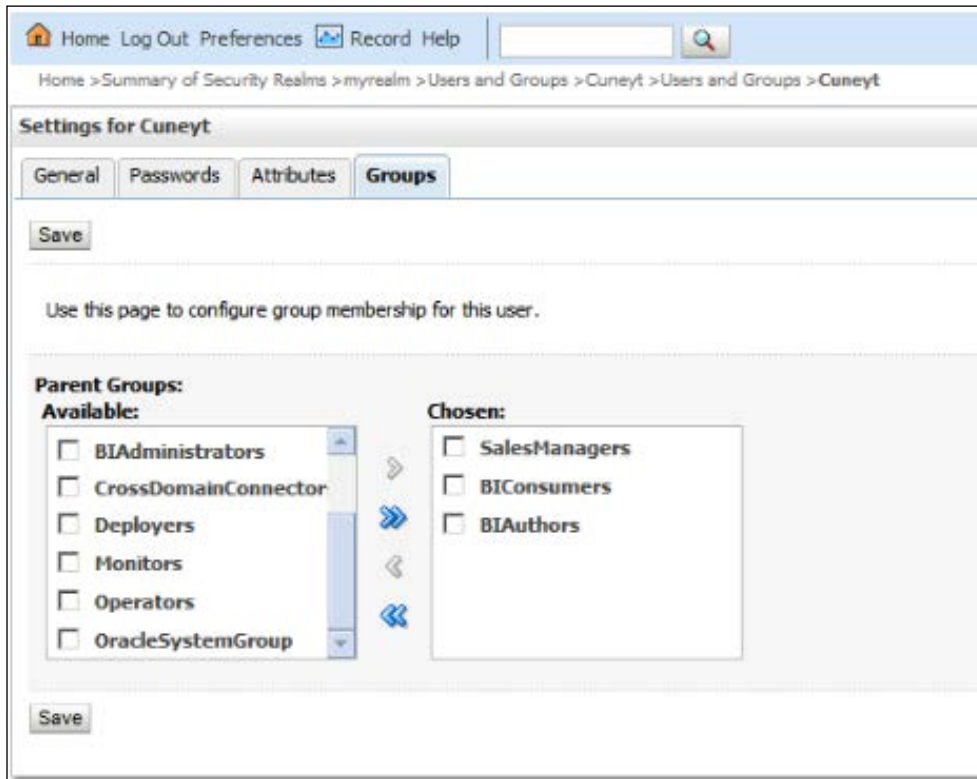
How it works...

We've added the user into these groups:

▸ `SalesManagers`: This is the sample group we've created. We're going to use this group in our sample scenario.

- ▸ `BIConsumers`: This is a default group and it's mapped to the `BIConsumer` application role in the Enterprise Manager. The members of this application role can access the analyses in Presentation Services. End-users should be a member of this group to access the analyses that are created by the BI developers.

- ▸ `BIAuthors`: This is another default group and it's mapped to the `BIAuthor` application role. The members of this role can access the analyses and also they can create new analyses. The users who want to construct the analyses should be a member of this group. Power users and BI developer roles are the candidates for the `BIAuthors` group.

There's more...

Now, we'll have to map these groups to application roles in order to set up permissions in the repository. These groups are only used to categorize the user accounts in the WebLogic Server. Only application roles are going to be displayed in the **Identity Manager**. We're going to learn how to configure the application roles in the next recipe.

Creating application roles

Application roles define a set of permissions that are granted to a user or group and they are defined in **Oracle Enterprise Manager Fusion Middleware Control**. These are the default application roles:

- ▸ `BIAdministrator`: This application role grants administrative permissions. This application role is a member of the `BIAuthor` role. So it inherits the permissions from the `BIAuthor` role.

    - ❏ `BIAuthor`: This application role grants permissions to create and edit analyses. This role is a member of `BIConsumer` role.

        - ❏ `BIConsumer`: This role grants permissions to access analyses that are created by other users.

- ▸ `BISystem`: This role is required by the Oracle BI system

How to do it...
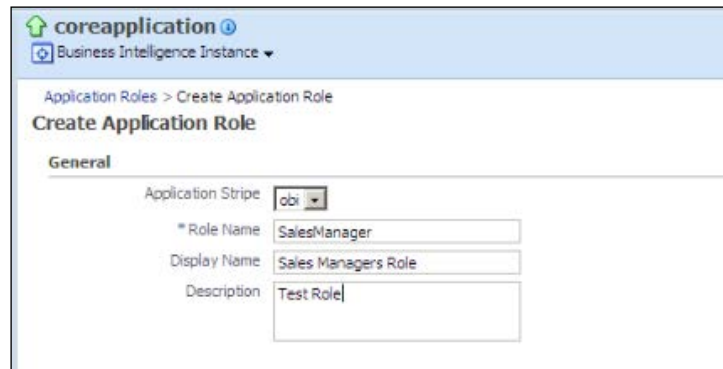
1. Log in to **Oracle Enterprise Manager Fusion Middleware Control**. Right-click on `coreapplication` and select **Application Roles** from the **Security** menu.
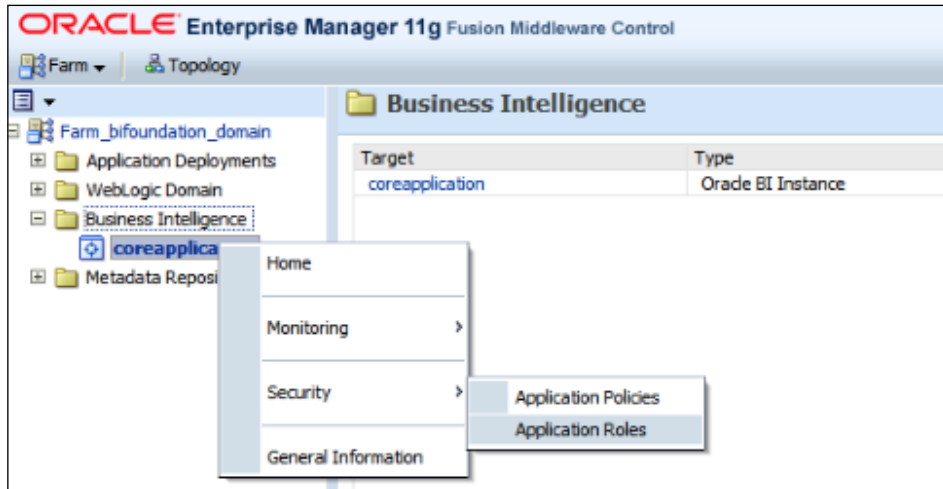
2. You'll see the list of existing **Application Roles**. Click on the **Create** button to start creating a new application role named `SalesManager`.
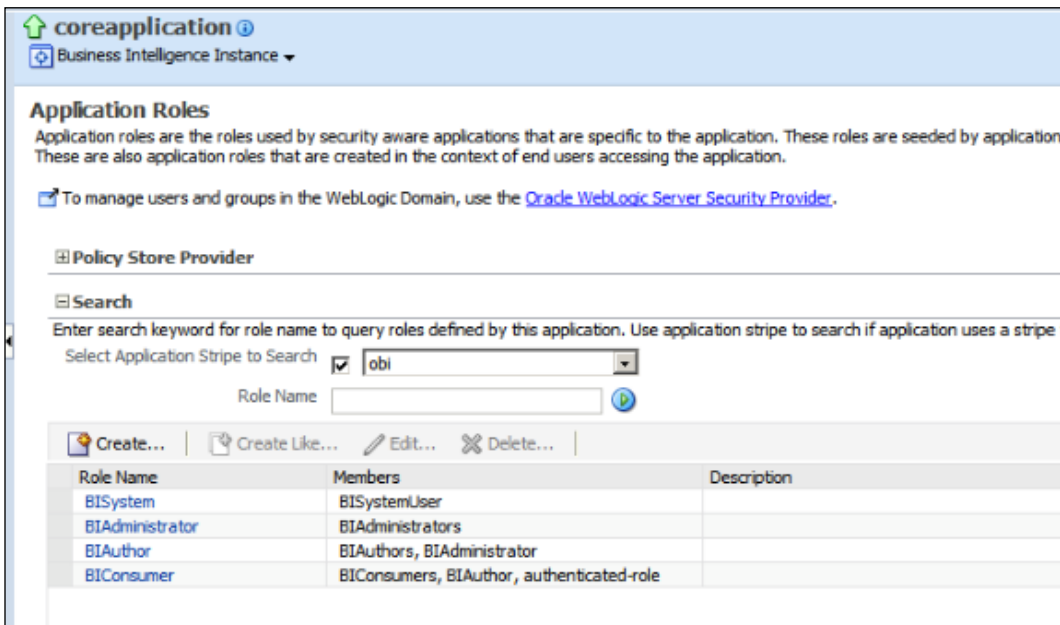
3. Enter the name and display name of the new application role. We're going to use `SalesManager` as the name.
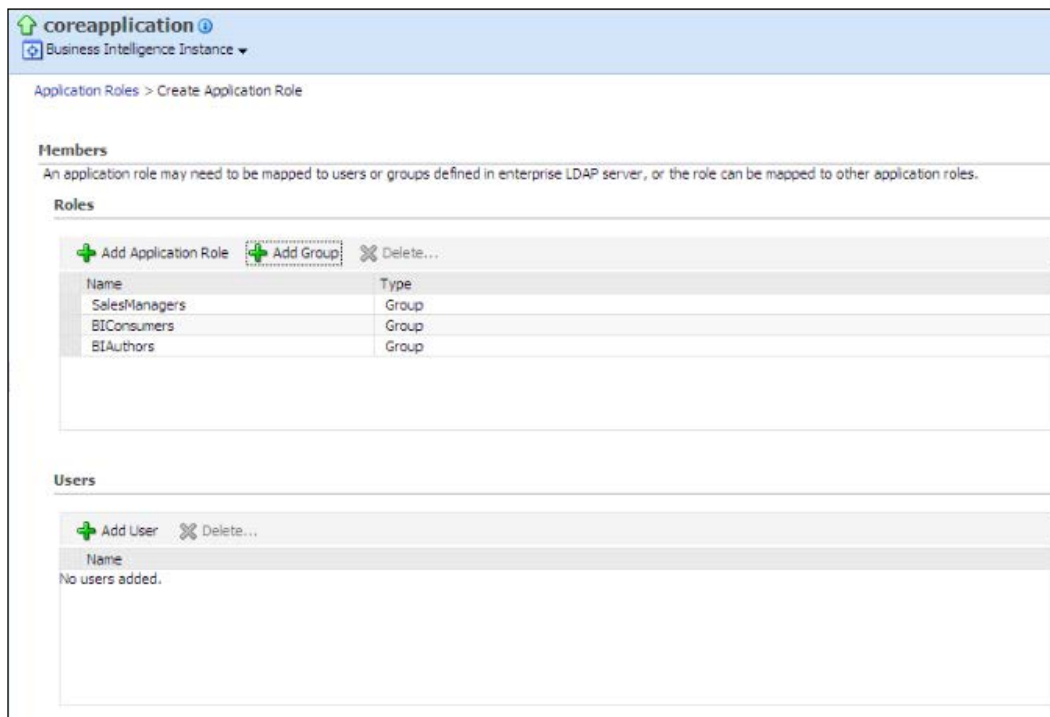
4. When you scroll down, you'll see the **Members** section. You can map groups or users to this application role. Click on the **Add Group** button to select groups and click on **OK** to finish creating the `SalesManager` application role.

5. After creating the application role, we're going to restart the BI Server component to make the new application roles available. Click on the **Restart** button in the **Overview** tab.
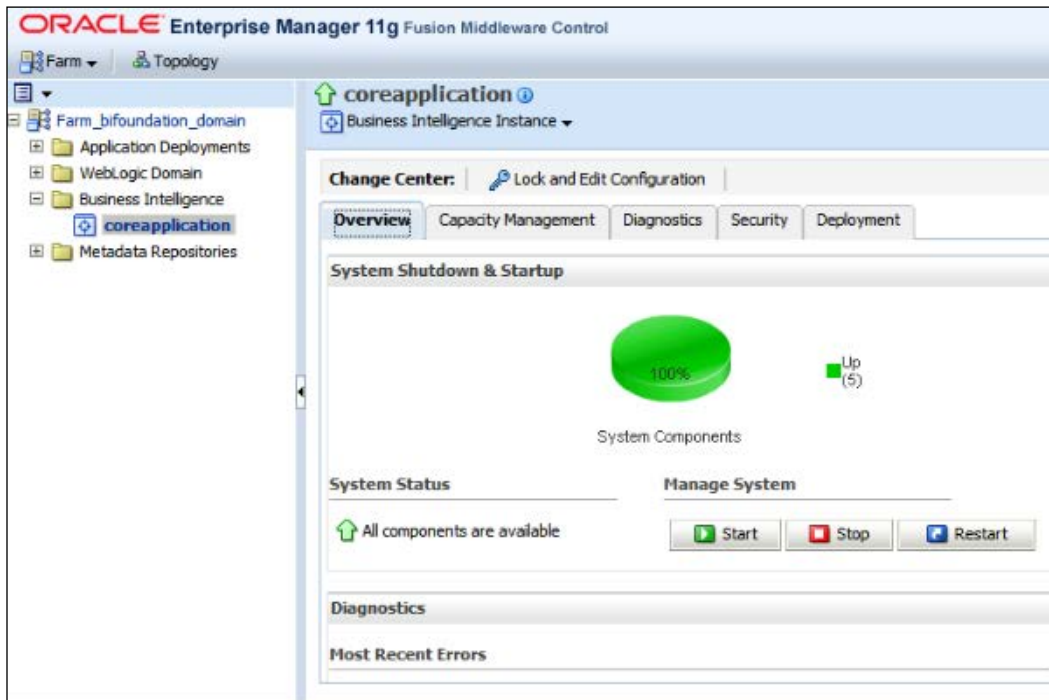
How it works...

We've created the `SalesManager` application role and mapped it to `SalesManagers` group. So the user account that is a member of the `SalesManagers` group is going to inherit the security permissions from the application role.

There's more...

The users, groups, and application roles are all created. Now it's time to use them, in order to set up permissions. These identity objects won't be displayed by default in the repository. They should be imported to the repository. We're going to learn how to set the object permissions by using these application roles in the next recipe.

Setting up permissions on repository objects

After creation of the user accounts and the application roles, we're going to discuss setting up permissions on the repository objects. Business users can access only the Presentation layer objects in the repository through the Presentation Services. All of the authenticated users shouldn't be able to access all subjects areas. They should see the objects that they need to see. We're going to learn how to set up permissions on the Presentation layer objects in this recipe.

We can setup the permissions on these objects:

- ▸ Subject areas
- ▸ Presentation tables
- ▸ Presentation columns

Getting ready

Users are going to be imported to the repository. To achieve this, we'll have to open **BI Administration Tool** and access the repository in online mode. We can't import these objects in offline mode.

How to do it...

1. Open the **BI Administration Tool** and open the repository in online mode.
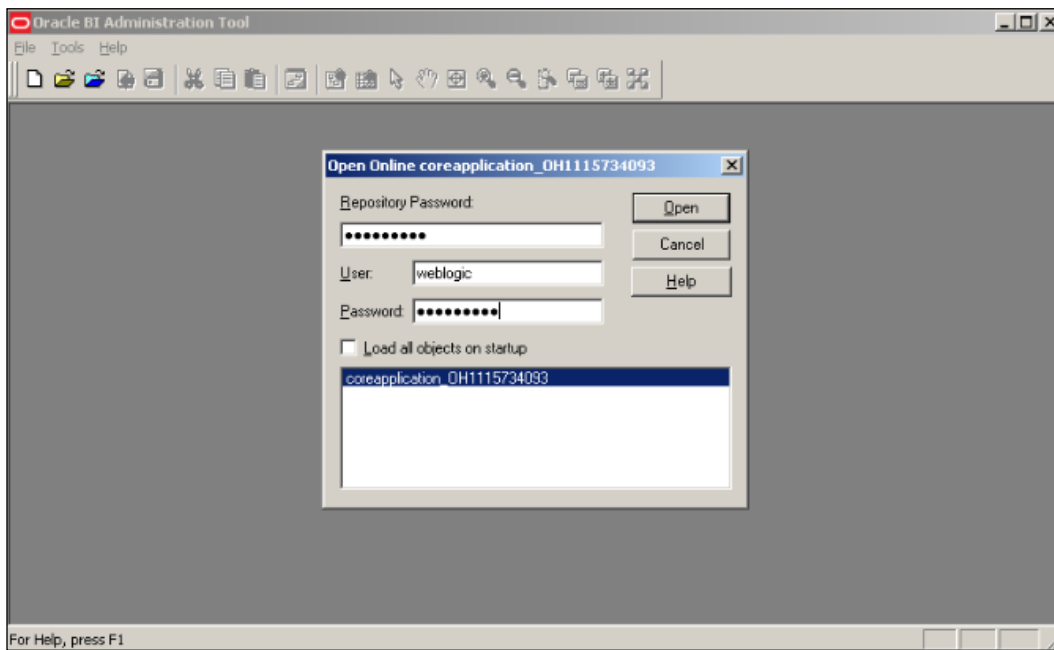
2. Open the **Identity Manager** from **Manage** menu and check the list of the existing users. You'll only see the `weblogic` user account.
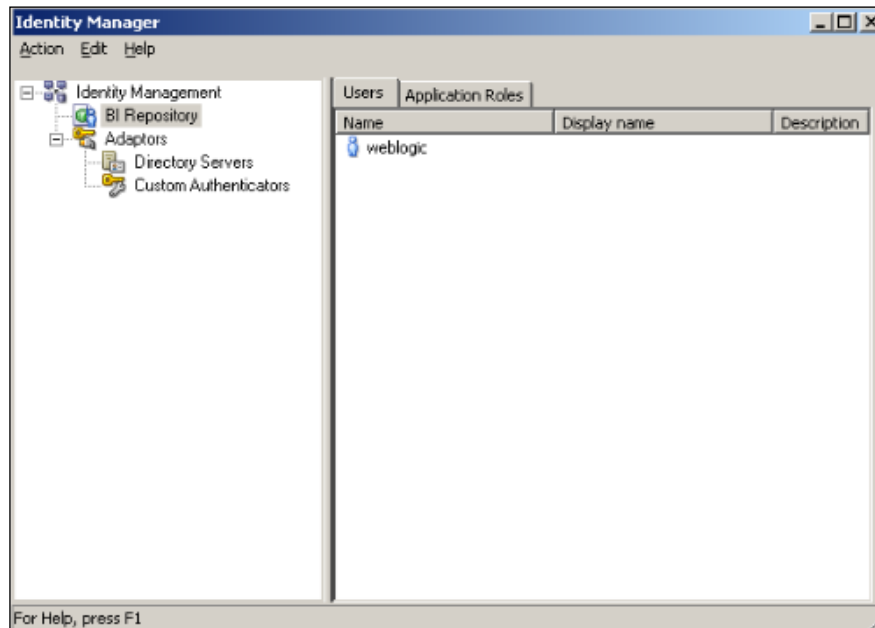
3. When you click on the **Application Roles** tab, you should see all the default application roles, including the `SalesManager` role.
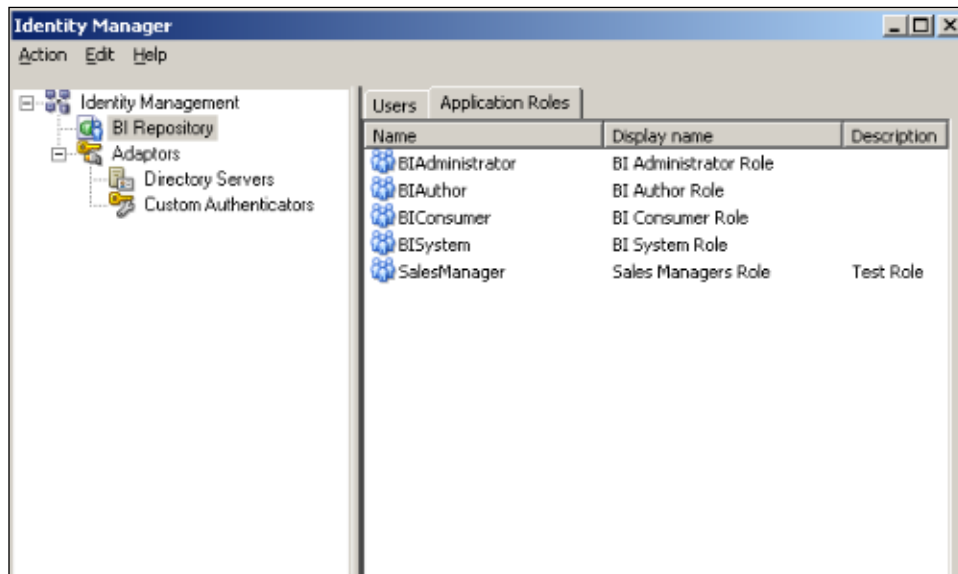
4. Click on the **Set Online User Filter...** option from the **Action** menu.
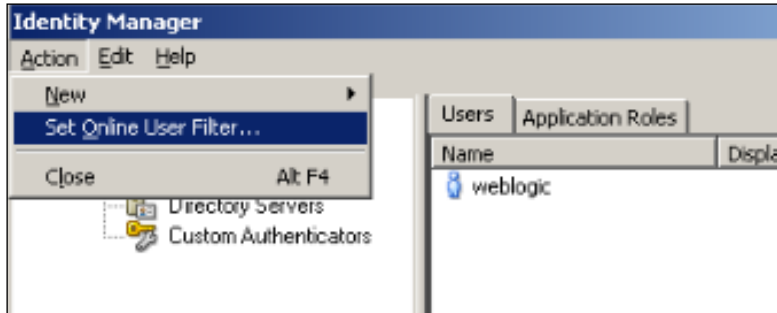
5. Enter * as a wild character to import all user identities.
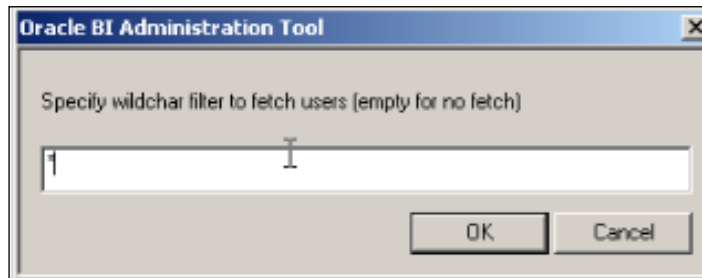
6. You'll see all the user accounts. Then close the **Identity Manager** and return to **BI Administration Tool**.
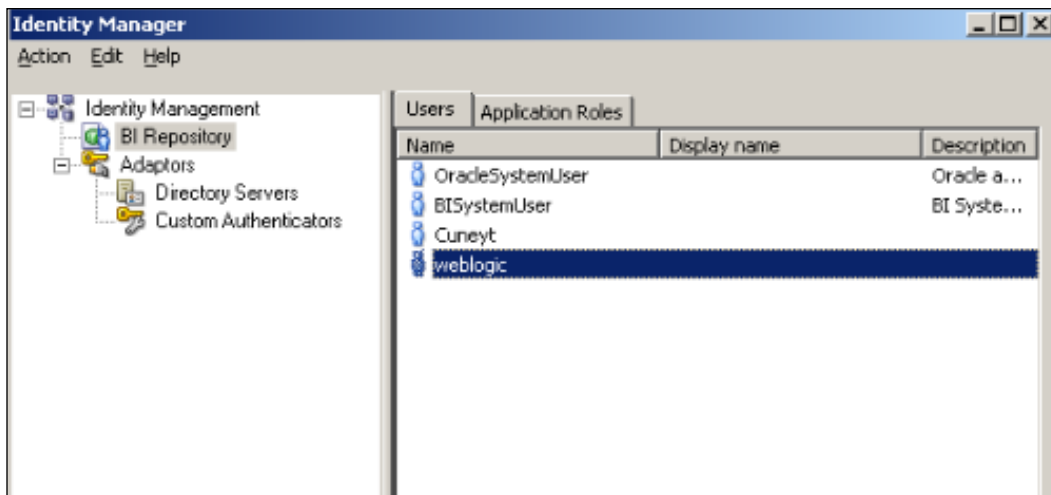
7. Double-click on `Sales` subject area and open the properties window. You'll see the **Permissions** button. The same button exists on all presentation tables as well. You can set up permissions on subject area or also on each presentation table.
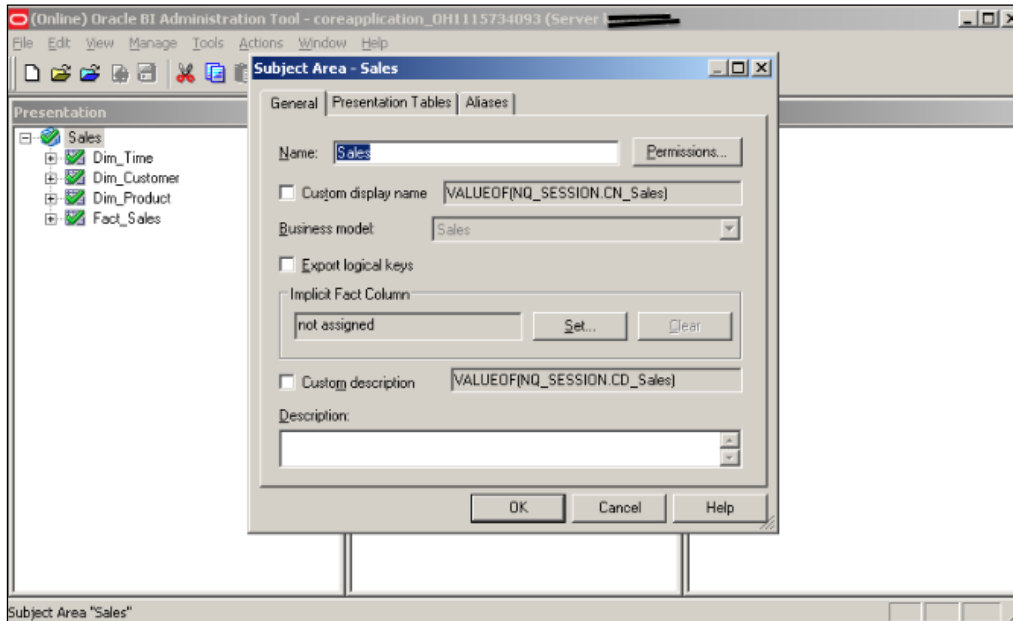
8. A permissions window will pop up. You'll see the list of applications roles and users as well. You can change the default permissions from this window. You can easily limit access to some application roles or users. The `Read/Write` permission is going to be needed only when you enable end-users to gain benefit of the write-back feature.
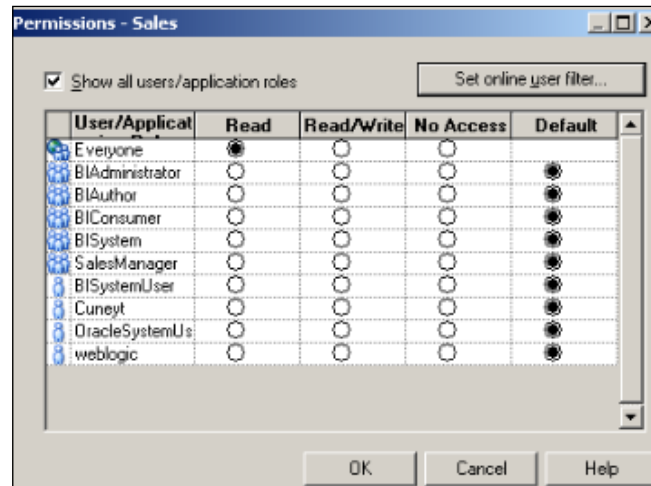
How it works...

We've configured the permission settings on the repository objects. When the business users log in to Presentation Services, they are going to see the subject areas based on the permissions that are granted on them. If a user doesn't have permission on a specific subject area then that subject area will not be displayed in Presentation Services.

There's more...

These are all permissions regarding the repository objects. You might be interested in row-level restrictions. We're going to cover row-level security in the *Creating Data Filters* recipe later in this chapter.

Configuring query limits

Configuring limits are very useful when it comes to performance tuning. End-users might execute queries and the returning result set might contain hundred thousands of rows. Eventually, this will negatively impact the performance. We'll have to configure **query limits** to avoid such a situation.

How to do it...

1. Query limits can be configured at the user or the application role level. Applying query limits at the application role level will be easier. Open the **Identity Manager** and double-click on **Application Role**.
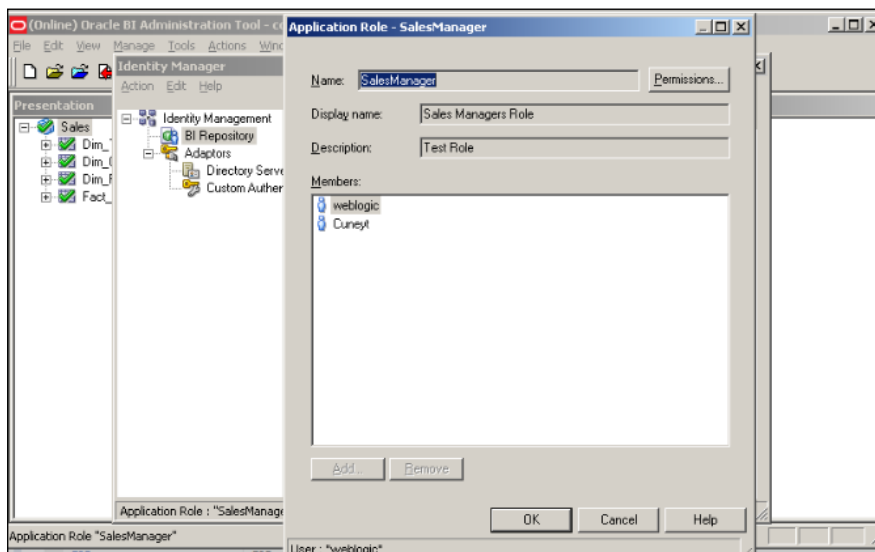
2. Click on the **Permissions** button. You'll see **User/Application Role Permissions** window. There are three tabs in this window. Click on the **Query Limits** tab.
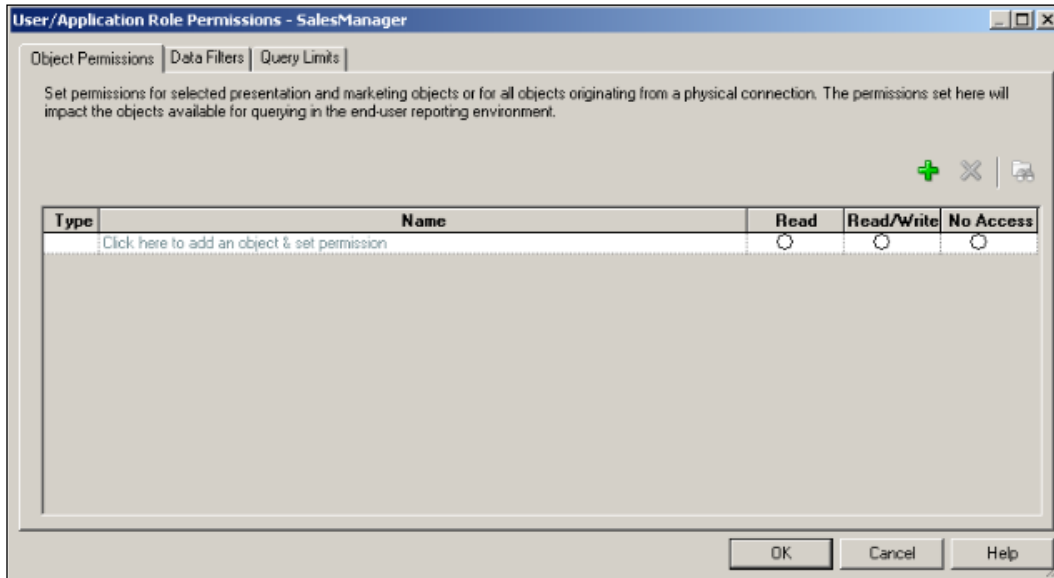
3. All query limits are ignored by default. One of the limits is called as `Status Max Rows` and it's also ignored. The `Max Rows` attribute is set to `100000` by default.
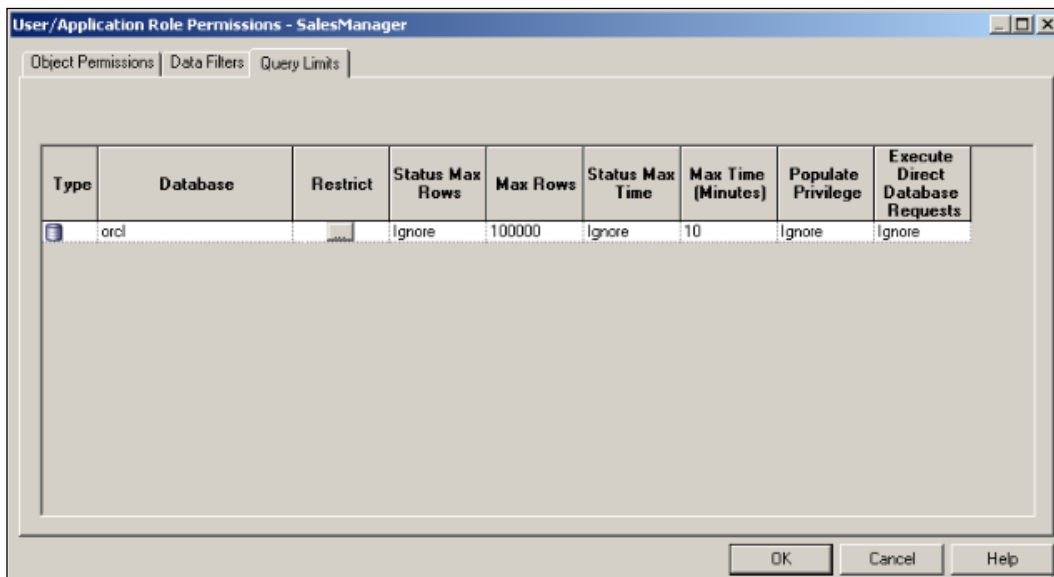
4. Now let's assume that we need a limit that the result set cannot exceed 1000 rows for the `SalesManager` application role. Change the `Max Rows` value to `1000` and click on the `Status Max Rows` column. Change it to **Enable**.

How it works...

After configuring these limits, business users will not be able to execute all kind of queries. Once the logical request is generated, BI Server is going to check these limits. Depending on the limit values, business users might see error messages. These limits could be applied with different values to different database connections. In our case, there's only one database connection and its name is `orcl`.

You might also apply a limit regarding the duration of the query.

There's more...

Executing **direct database requests** means end-users could create their own query regardless of subject areas and execute the physical query against the specific database. But this could be risky. So you can also disable this setting for databases.

Specifying the time restrictions

Introduction

Limiting access to the BI Server during the maintenance period could be a good solution. You can apply time-based restrictions for users or application roles. You can easily disallow business users to execute queries at the specified periods. We're going to learn how to implement time-based restrictions in this recipe.

How to do it...

1. To implement time-based restrictions double-click on the **User** or **Application Role** tab and go to the **Query Limits** tab. Click on the `Restrict` column in this window.
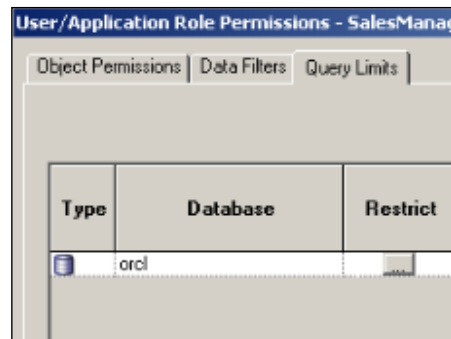
2. You can easily set the allowed or disallowed periods by clicking on the **Allow** and **Disallow** buttons.
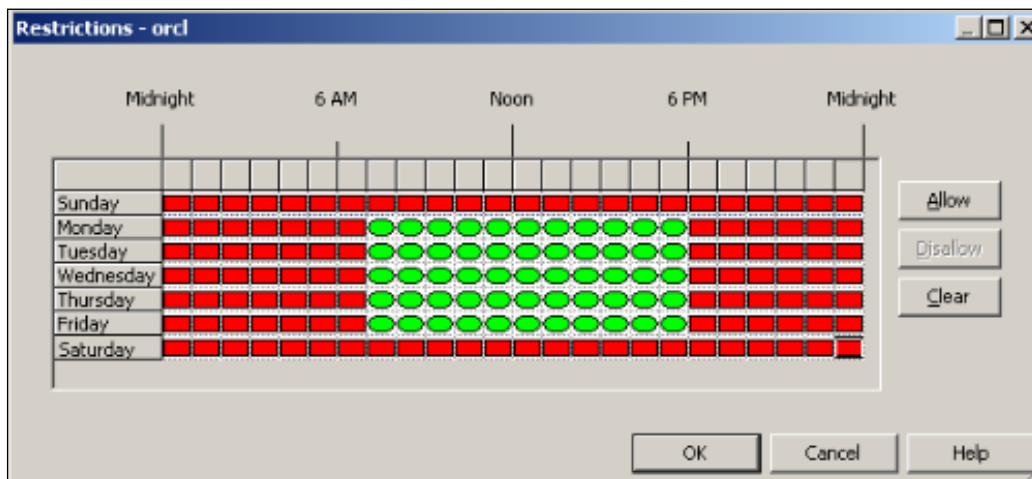
How it works...

These time-based restrictions can be customized for each database and also for different application roles. For example, managers could log in during non-business hours and others couldn't.

Creating data filters

End-users might need to access the same presentation tables but they're required to see different data sets. This could be achieved only by creating data filters. This is also called row-level security. Let's assume that two different users need to access the same table. But one of the users may be responsible for the west region and the other may be the employee who is responsible for the east region. They should access only the data that is related to their region. In such cases, row-level security will be the best solution.

We're going to learn how to implement data filters in this recipe.

How to do it...

1. Open the **Identity Manager** and double-click on any application role. When the **User/Application Role Permissions** window pops up, click on the **Permissions** button and go to the **Data Filters** tab. You'll notice that there's no data filter by default.
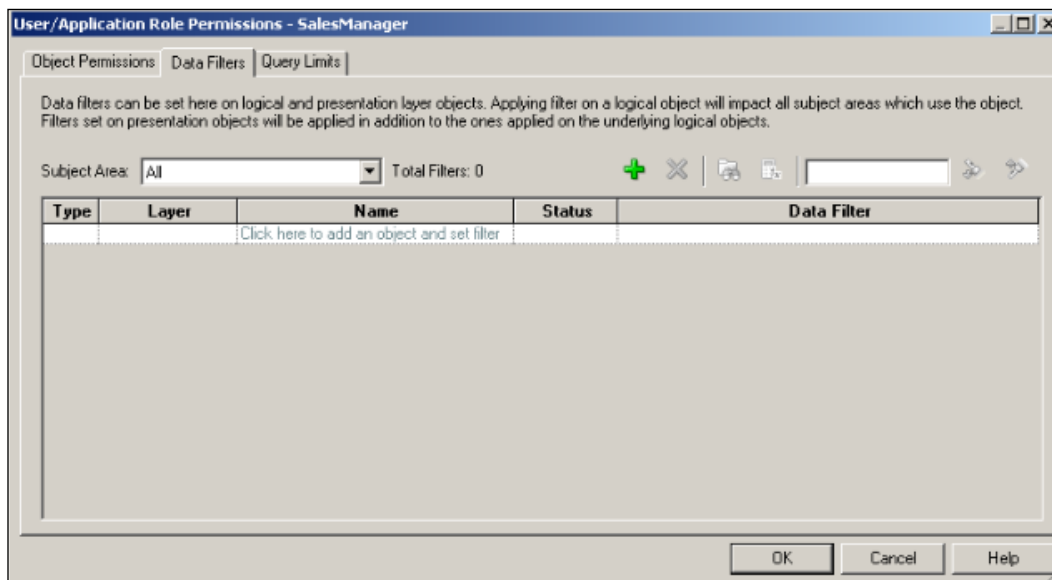
2. Click on the **Add** button to create a data filter for the `SalesManager` application role. The **Browse** window will pop up and you'll select a presentation column to apply the filter on. In our case, we're going to implement a data filter based on the `Region` column.
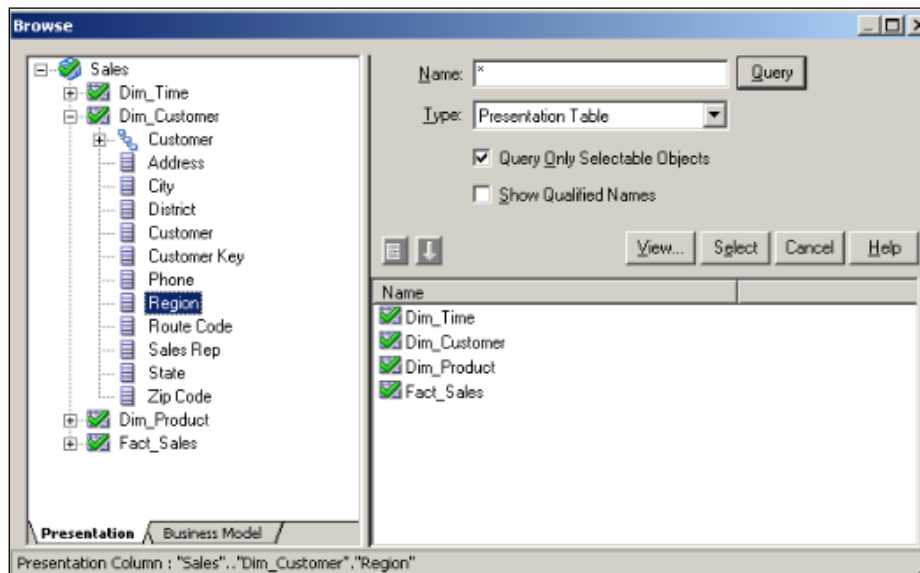
3. Select the `Presentation` column. It is going to be added to the data filter automatically. Now click on the Edit Expression icon to specify the filter condition.
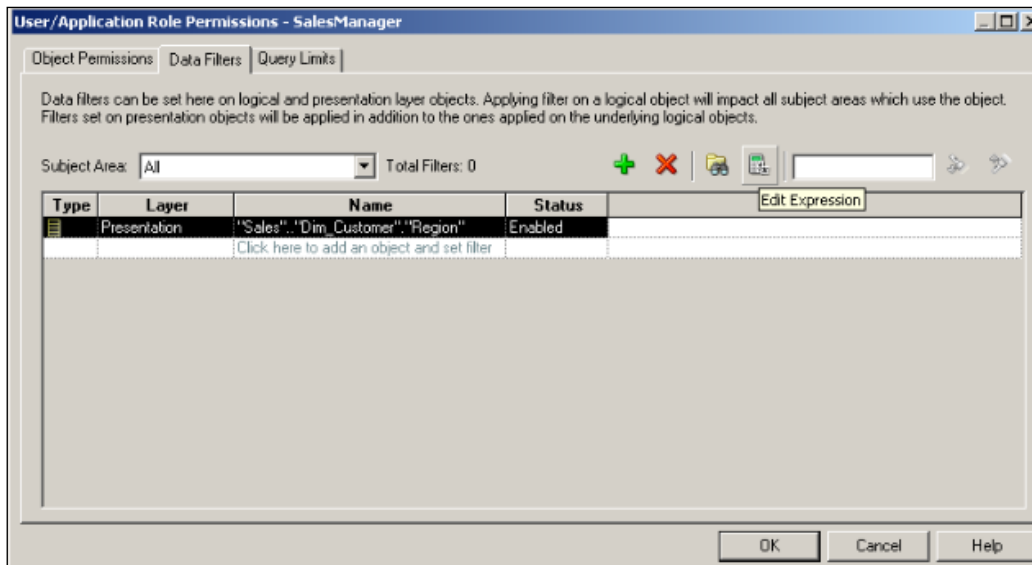
4. Use the **Expression Builder** window to create a data filter. Click on **OK** to close the **Expression Builder**.
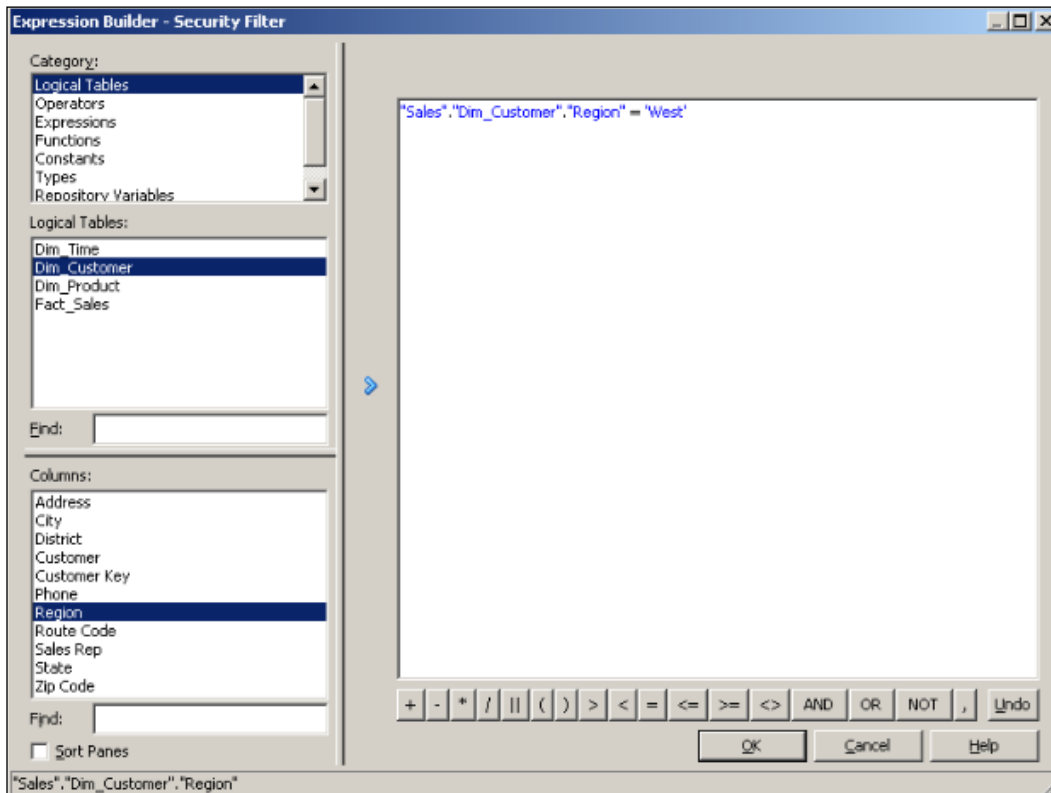
5. Here is the result. We have created the data filter based on the `Region` column. The result set is going to be generated with this filter.
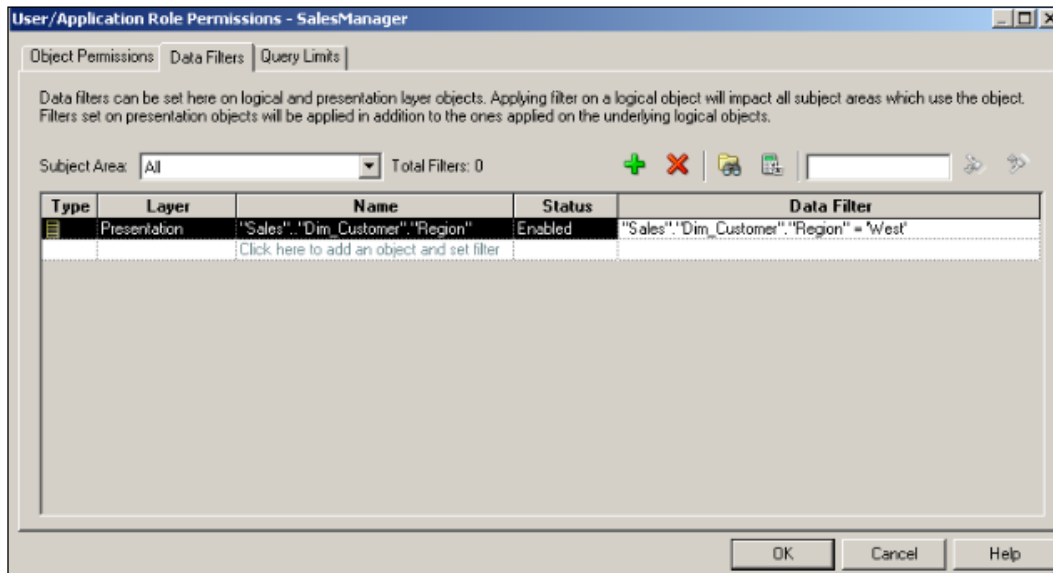
How it works...

After creating this data filter if anyone who is member of the `SalesManager` application role tries to execute any query, the result set is going to be generated, but a filter is going to be applied automatically. Only the records belonging to the west region will be displayed. In our sample scenario, we've used a static value as a condition.

When it comes to creating data filters, you don't have to specify a filter condition by using a string. We might gain benefit by using variables. Session variables could be used instead of the string value. Let's assume that we've already created a session variable named `VAR_REGION`. Then we would have used the filter as specified below:

```
"Sales"."Dim_Customer"."Region" = VALUEOF(NQ_SESSION."VAR_REGION")
```

6
Managing Usage Tracking and Enabling the Cache

In this chapter, we will cover:

- ▶ Enabling Usage Tracking
- ▶ Creating the Business Model for Usage Tracking
- ▶ Creating dashboards for Usage Statistics
- ▶ Enabling the cache
- ▶ Gathering cache statistics
- ▶ Managing the cache

Introduction

When Oracle BI is deployed, we can't be sure about what kind of queries are going to be executed by the end users. We need to know end users' behaviors. Another business challenge can be regulatory requirements that include monitoring **usage statistics**. We already learned about query logging. When we change the logging attribute of any user, we know that the query is going to be logged in the `NQQuery.log` file. But it doesn't satisfy all business requirements. Instead of managing the logging attribute for each user, we can gain the benefit of **Usage Tracking**.

Enabling Usage Tracking
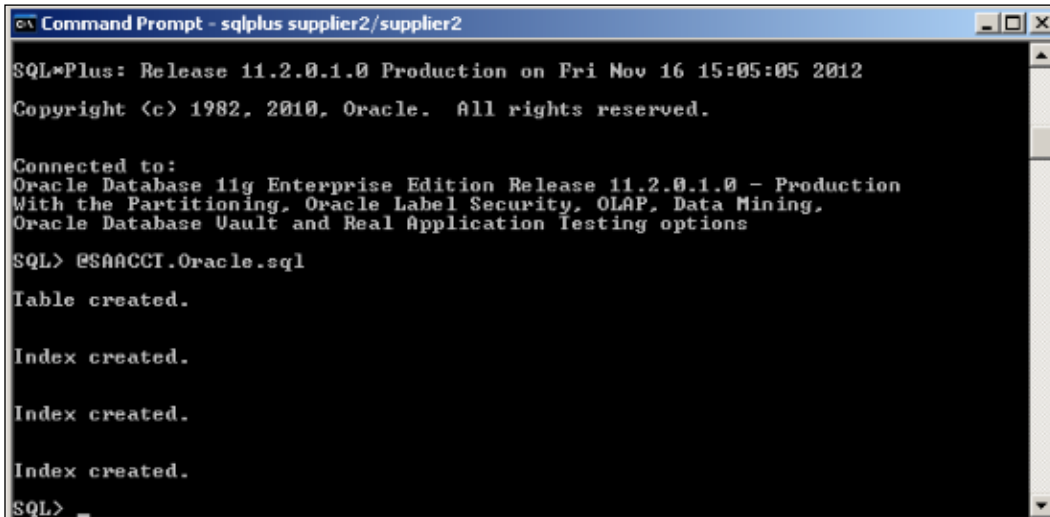
Usage Tracking is a very important feature of Oracle BI. It can be enabled at the server level and it can store all query execution information in a table in the database. This feature allows us to create usage reports as well. We're going to learn how to enable Usage Tracking in this recipe. After enabling Usage Tracking we're going to access some sample analyses. This will generate the logs about the query executions and these logs will be stored in the Usage Tracking table. Then we will create the reports that will show the usage statistics.

How to do it...

1. To store the usage data in a database table, first of all we'll have to create this table. The table creation script can be found in the installation directory. The folder where you can find the scripts is `<ORACLE_INSTANCE>\bifoundation\ OracleBIServerComponent\coreapplication_obis1\schema`.
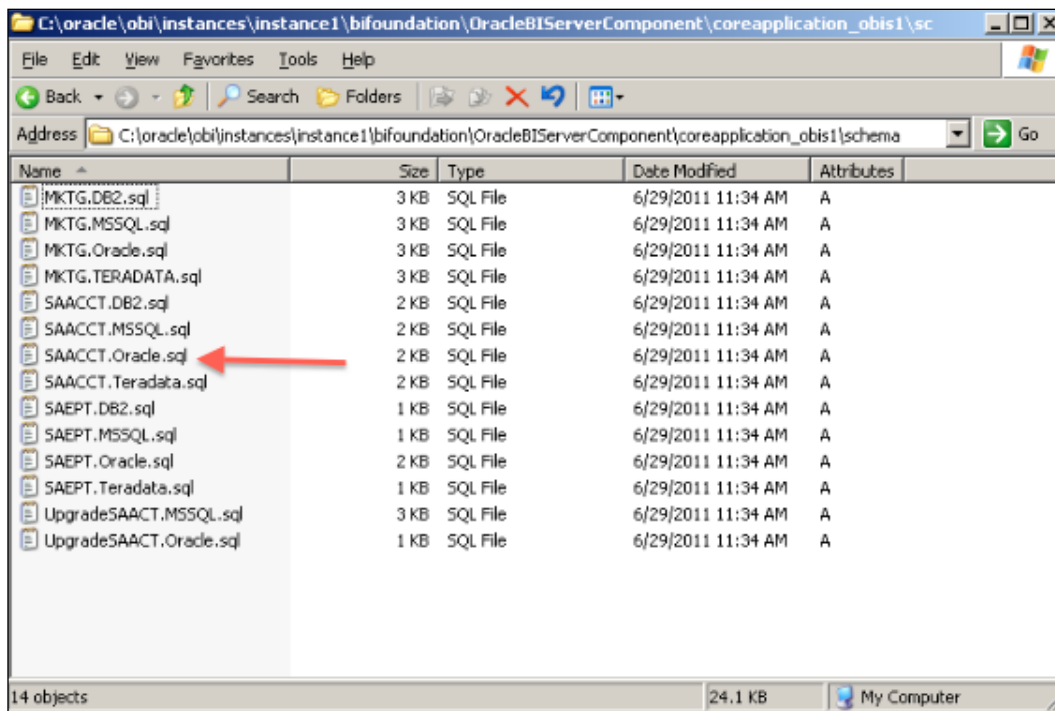
2. The `SAACCT.Oracle.sql` script file is going to be used to create the Usage Tracking table. It's good to create this table in another schema but we're going to use the same schema, named `SUPPLIER2` in the example.

3. Now we're going to import this table but it'll be good to create a new database in the repository. Click on **New Database...**.
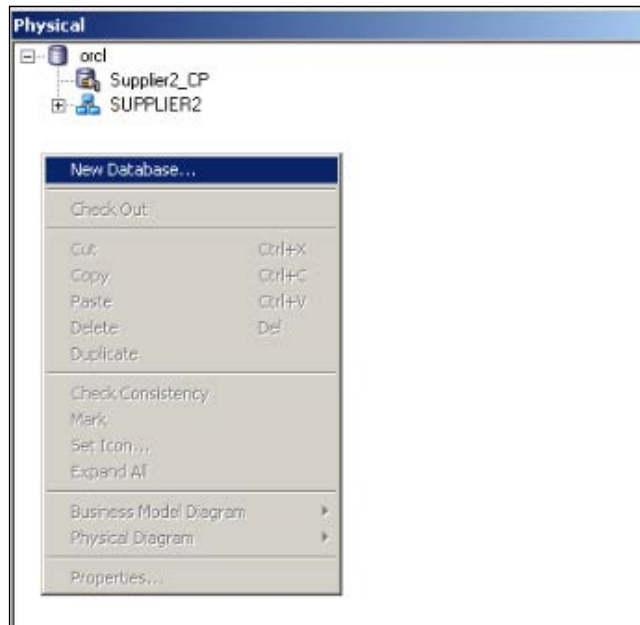
4. We enter the name of the database and select the type of the database from the **Database** drop-down list in the **Database – UsageTrackingDB** window.
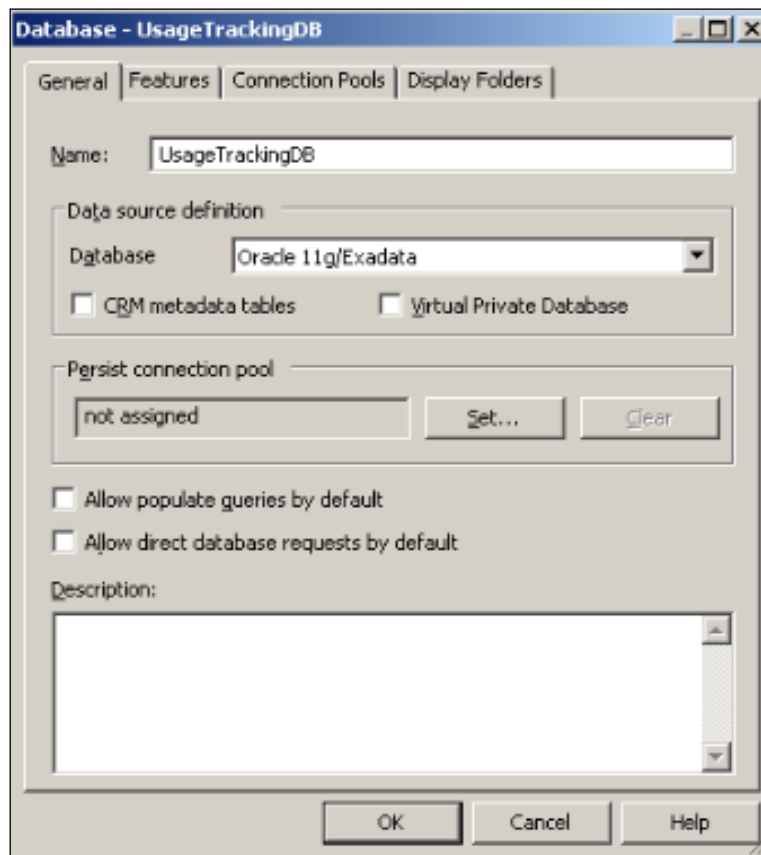
5. Next step will be creation of the connection pool object for the new database. Right-click on the database and select **New Object** and then **Connection Pool...**.
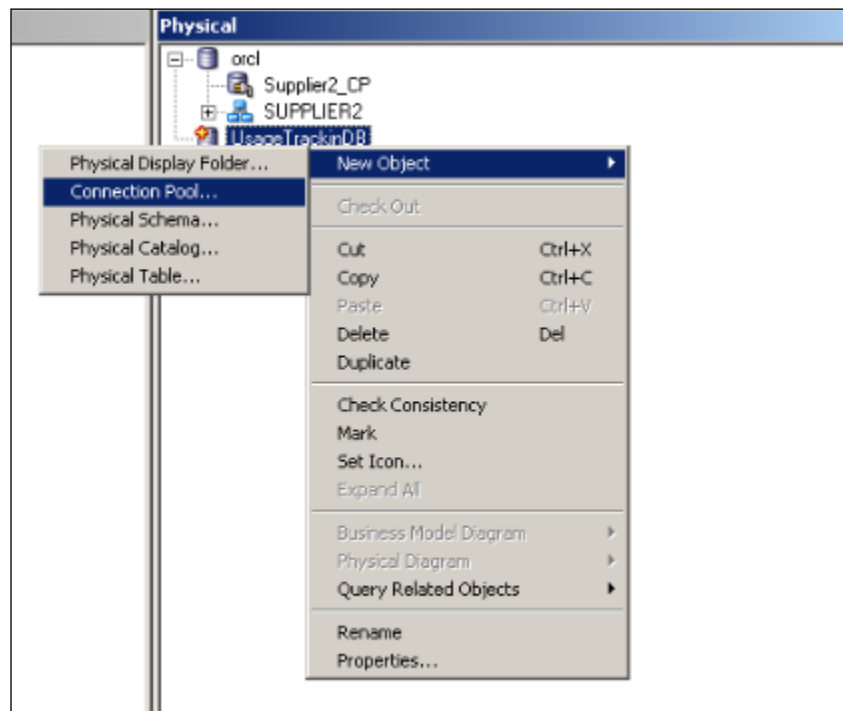
6. Enter the details of the connection pool:

    ❑ **Name**: `UsageTrackingCP`

    ❑ **Call interface**: **Default (OCI 10g/11g)**

    ❑ **Data source name**: `orcl`

    ❑ **User name**: `supplier2`

❑ **Password**: `supplier2`
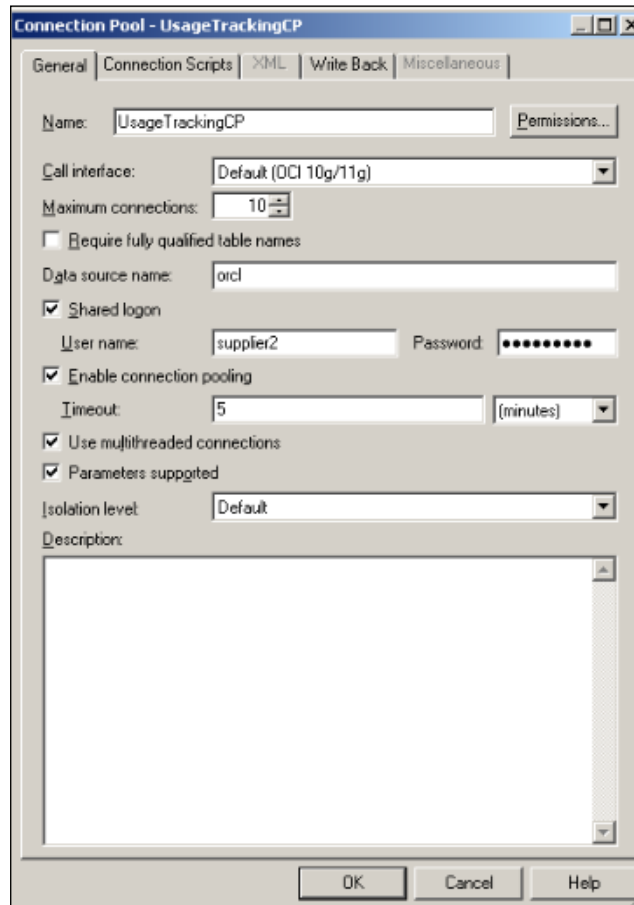
7. After creating the connection pool, we're going to import the new table, named `S_NQ_ACCT`. You'll see the imported table in the following screenshot. This Usage Tracking table consists of several columns. During the execution of the queries, the BI Server captures important information that will be very useful in monitoring the users' behaviors. Some of the columns are listed as follows:

   ❑ `CACHE_IND_FLG`: Indicates if the query is satisfied from the cache or not

   ❑ `QUERY_TEXT`: Stores the SQL statement that is executed

   ❑ `START_DT and START_HOUR_MIN`: Shows the start time in days, hours, and minutes of the query execution

❑ `END_DT and END_HOUR_MIN`: Shows the end time in days, hours, and minutes of the query execution

❑ `SAW_DASHBOARD and SAW_DASHBOARD_PG`: The dashboard and the dashboard page name of the analysis that is displayed

Physical
- ⊞ orcl
- ⊟ UsageTrackingDB
  - UsageTrackingCP
  - ⊟ SUPPLIER2
    - ⊟ S_NQ_ACCT
      - CACHE_IND_FLG
      - COMPILE_TIME_SEC
      - CUM_DB_TIME_SEC
      - CUM_NUM_DB_ROW
      - END_DT
      - END_HOUR_MIN
      - END_TS
      - ERROR_TEXT
      - IMPERSONATOR_USER_NAME
      - NODE_ID
      - NUM_CACHE_HITS
      - NUM_CACHE_INSERTED
      - NUM_DB_QUERY
      - PRESENTATION_NAME
      - QUERY_BLOB
      - QUERY_KEY
      - QUERY_SRC_CD
      - QUERY_TEXT
      - REPOSITORY_NAME
      - ROW_COUNT
      - SAW_DASHBOARD
      - SAW_DASHBOARD_PG
      - SAW_SRC_PATH
      - START_DT
      - START_HOUR_MIN
      - START_TS
      - SUBJECT_AREA_NAME
      - SUCCESS_FLG
      - TOTAL_TIME_SEC
      - USER_NAME

We've finished the repository steps. Now we're going to modify the configuration file Oracle BI Server in order to enable Usage Tracking. The configuration file path is `<ORACLE_INSTANCE>\config\OracleBIServerComponent\coreapplication_obis1` and the name of the file is `NQSConfig.INI`.

1. Now, we're going to edit this file. The configuration file is divided into sections. One of the sections is named `[USAGE_TRACKING]`.

   ```
   [USAGE_TRACKING]
   ENABLE = NO;
   STORAGE_DIRECTORY = "<directory path>";
   CHECKPOINT_INTERVAL_MINUTES = 5;
   FILE_ROLLOVER_INTERVAL_MINUTES = 30;
   CODE_PAGE = "ANSI";  # ANSI, UTF8, 1252, etc.
   DIRECT_INSERT = YES;
   PHYSICAL_TABLE_NAME = "<Database>"."<Schema>"."<Table>";
   CONNECTION_POOL = "<Database>"."<Connection Pool>";
   BUFFER_SIZE = 250 MB;
   BUFFER_TIME_LIMIT_SECONDS = 5;
   NUM_INSERT_THREADS = 5;
   MAX_INSERTS_PER_TRANSACTION = 1;
   ```

2. We're going to modify some attribute values in this section. The modified values are specified in bold. If the `DIRECT_INSERT` parameter is set to `NO`, the usage tracking statistics will be stored in the files instead of the tables. Recommended practice is setting this value to `YES`. When it's set to `YES`, you will have to specify the `PHYSICAL_TABLE_NAME` and the `CONNECTION_POOL` parameter values. These parameters set the storage for the statistics. The `BUFFER_SIZE` parameter sets the memory size that will be allocated by the BI Server for the usage tracking. Depending on the workload, you may increase this value.

   ```
   [USAGE_TRACKING]
   ENABLE = YES;
   STORAGE_DIRECTORY = "<directory path>";
   CHECKPOINT_INTERVAL_MINUTES = 5;
   FILE_ROLLOVER_INTERVAL_MINUTES = 30;
   CODE_PAGE = "ANSI";  # ANSI, UTF8, 1252, etc.
   DIRECT_INSERT = YES;
   ```

```
PHYSICAL_TABLE_NAME =
"UsageTrackingDB"."SUPPLIER2"."S_NQ_ACCT";
CONNECTION_POOL = "UsageTrackingDB"."UsageTrackingCP";
BUFFER_SIZE = 250 MB;
BUFFER_TIME_LIMIT_SECONDS = 5;
NUM_INSERT_THREADS = 5;
MAX_INSERTS_PER_TRANSACTION = 1;
```

3. After saving the configuration file, we'll have to restart the BI Server by using Oracle Fusion Middleware Enterprise Manager. Click on the **Restart** button.

How it works...

1. It's time to test **Usage Tracking**. We're going to log in to Presentation Services and create a simple analysis, as in the following screenshot:

2. You'll see that log data about the query execution is inserted into the `S_NQ_ACCT` table.

There's more...

After enabling Usage Tracking, we can easily learn about the user behavior and these important statistics data will be used in performance tuning. We're going to focus on only frequently executed queries to make decisions in the design of the aggregate tables. You can see the usage tracking information that is accessed through Presentation Services in the following screenshot:

USER_NAME	QUERY_TEXT	START_HOUR_MIN	QUERY_TEXT	ROW_COUNT	TOTAL_TIME_SEC
weblogic	SELECT 0 s_0, "Sales"."Dim_Customer"."Region" s_1, "Sales"."Dim_Customer"."State" s_2, "Sales"."Fact_Sales"."Dollars" s_3 FROM "Sales" ORDER BY 1, 2 ASC NULLS LAST, 3 ASC NULLS LAST	09:56	1	34	0
weblogic		11:58	1	34	0
weblogic		16:51	1	34	0
weblogic		17:34	1	34	0
weblogic		20:44	1	34	1
weblogic	SELECT 0 s_0, "Sales"."Dim_Customer"."Region" s_1, "Sales"."Dim_Customer"."State" s_2, "Sales"."Fact_Sales"."Dollars" s_3 FROM "Sales" WHERE ("Dim_Customer"."Region" = 'East') ORDER BY 1, 2 ASC NULLS LAST, 3 ASC NULLS LAST	16:51	1	16	0
weblogic		17:36	1	16	0
weblogic	SELECT 0 s_0, "Sales"."Dim_Customer"."Region" s_1, "Sales"."Dim_Customer"."State" s_2, "Sales"."Fact_Sales"."Units Ordered" s_3 FROM "Sales" ORDER BY 1, 2 ASC NULLS LAST, 3 ASC NULLS LAST	21:50	1	34	0
weblogic	SELECT 0 s_0, "Sales"."Dim_Customer"."Region" s_1, "Sales"."Dim_Customer"."State" s_2, "Sales"."Fact_Sales"."Units Ordered" s_3, "Sales"."Fact_Sales"."Units Shipped" s_4 FROM "Sales" ORDER BY 1, 2 ASC NULLS LAST, 3 ASC NULLS LAST	21:20	1	34	1
weblogic	SELECT 0 s_0, "Sales"."Dim_Customer"."Region" s_1, "Sales"."Dim_Customer"."State" s_2, "Sales"."Fact_Sales"."Units Ordered" s_3, "Sales"."Fact_Sales"."Units Shipped" s_4 FROM "Sales" WHERE ("Dim_Customer"."Region" = 'East') ORDER BY 1, 2 ASC NULLS LAST, 3 ASC NULLS LAST	21:21	1	16	0

Creating the Business Model for Usage Tracking

Usage Tracking is enabled now. The logs are going to be inserted into the log table always because we've set the `DIRECT_INSERT` parameter value to `YES` in the configuration file. We can execute queries against this table and check the statistics. If you have a physical table in the repository, it'll be easier to access the statistics through Presentation Services. As you'll remember, in order to access any physical table from the Web UI, we have to create a Business Model and also a Subject Area for the model. After that, we can access this important data from Presentation Services.

How to do it...

1. We're going to start creating a new Business Model. But this time, there's only one Physical table. This is the reason why we don't need any physical join. We're going to create four logical tables and define logical joins in the Business Model and Mapping layer. Right-click on the BMM layer, select **New Business Model**, and enter the name of the Business Model.

2. We're going to create four logical tables in this new Business Model. So right-click on the model and select **New Object** and then **Logical Table**.

3. You'll just enter the name of the logical table such as `Measures`.

4. The remaining three logical tables are going to be done the same way. You can see them in the following screenshot:

5. The logical tables are not mapped to any Physical table so we're going to add the logical table source to each of the logical tables. We're going to drag-and-drop corresponding physical columns onto the logical tables. The `UsageTracking` Business Model is shown as follows:

❑ `Content` logical table: `QUERY_TEXT, REPOSITORY_NAME, SUBJECT_AREA_NAME`

❑ `Measures` logical table: `QUERY_TEXT, ROW_COUNT, TOTAL_TIME_SEC`

❑ `Time` logical table: `START_DT, START_HOUR_MIN, END_HOUR_MIN`

❑ `User` logical table: `USER_NAME`

6. In the next step, we'll have to define the relations between the logical tables by using Business Model Diagram. After opening the **Business Model Diagram** window, click on the **New Join** button. Next, click on first measure logical table, then the dimension table. You're going to repeat this three times so that three joins will be created. Additionally, we need to define the logical keys on the logical tables like the following:

      ❑ **Content**: `QUERY_TEXT`

      ❑ **Time**: `START_DT`

      ❑ **User**: `USER_NAME`

7. The last step regarding the BMM layer is going to be about defining aggregation rules of the logical columns in the `Measure` logical table.

8. All of the tasks in the BMM layer are finished now. Just to make this new Business Model visible in Presentation Service, we're going to create a Subject Area in the Presentation layer. We're going to achieve this by dragging-and-dropping the `UsageTracking` Business Model to the Presentation layer. Now, you can also save the repository and check the global consistency. You'll notice that the `UsageTracking` Business Model will be set to `enabled`.

How it works...

After enabling the Business Model, we can access this **UsageTracking** subject area from Presentation Services for reporting purposes.

Creating dashboards for Usage Statistics

Oracle BI Server's repository is now ready to create reports regarding **Usage Statistics**. We're going to create some sample analyses and publish them in the dashboard for BI Administrators. Creating dashboards is going to be covered in *Chapter 10*, *Creating and Configuring Dashboards*, and we're going to use an existing dashboard. In our sample scenario, it will be **My Dashboard**. Obviously, this **UsageTracking** subject area should not be accessible by the end users. This is only for BI Administrators.

How to do it...

1. We're going to create some analyses in the **UsageTracking** subject area. You can easily repeat this task to create multiple analyses. After logging in to Presentation Services, click on the **New** button and select **Analysis**. Then we're going to select the **UsageTracking** subject area to create the analysis. In the first example, I've only used four columns:

    □   SUBJECT_AREA_NAME

    □   START_DT

❑ `ROW_COUNT`

❑ `TOTAL_TIME_SEC`

2. When you click on the **Results** tab, you'll see the result set. In order to publish these analyses in the dashboard, we'll have to save the Web object by clicking on the **Save Analysis** button.

3. Here is another example of an analysis in the **UsageTracking** subject area.

4. And now it's time to publish these two analyses on a dashboard. BI Administrators will access the dashboard and see the statistics regarding query executions. Click on the **Dashboards** drop-down list and select **My Dashboard** from the menu list. Then we're going to click on the **Edit** hyperlink to open the **Dashboard Builder**.

5. When the Dashboard Builder is opened, we're going to drag-and-drop the previously saved analyses on to the **Drop Content Here** section, and then click on the **Save** button and then the **Run** button.

How it works...

We have added two new pieces of content into **My Dashboard**. When you go to the **My Dashboard** page, you'll see the analyses.

There's more...

This was a simple dashboard page. We're going to cover different views of the analyses in *Chapter 7, Creating Simple Oracle Business Intelligence Analyses*. We're going to add some graphic views to the existing analyses and this will make the reports more functional.

Enabling the cache

The queries that are executed against the data warehouses causes large amount of database processing. Every time we access an analysis, the query is going to be generated by the BI Server and it's executed against the database. In order to improve the performance, we should balance the workload by enabling the cache on the BI Server. So when the query is executed for the first time, the result set is going to be cached on the BI Server. The result set is going to be generated from the cache in the subsequent executions.

How to do it...

1. The BI Server's cache is going to be enabled from Fusion Middleware Enterprise Manager Control. You'll see the **Performance** property page under the **Capacity Management** tab. Then click on the **Lock And Edit Configuration** link to modify the values.

2. After selecting the **Cache Enabled** checkbox, you'll need to click on the **Activate Changes** link.

3. Changes are going to be activated and the **Restart to apply recent changes** link will appear. Click on that link and restart the BI Server.

How it works...

We have enabled the cache on the BI Server. After you execute queries, the generated result sets will be stored in the cache so that, in the subsequent execution of the same queries, the result is going to be retrieved from the cache. This is going to eliminate repeated database processing tasks.

There's more...

We also need to manage the cache because when the data warehouse is refreshed or updated, the cached should be maintained as well. Otherwise, end users will access old result sets that are satisfied from the cache. We're going to cover these challenges in the last task of this chapter.

Gathering cache statistics

Cache statistics can be gathered by using **Cache Manager**. It's a tool that is part of the BI Administration Tool. These statistics will help us to understand the queries that are satisfied from the cache. If most of the time the queries are not being retrieved from the cache, it means that we're not managing the cache successfully.

How to do it...

1. Open the **BI Administration Tool** in online mode. Cache Manager is not activated if you open the repository in offline mode. Click on the **Cache** menu item from the **Manage** menu list.

2. Cache Manager is going to be opened. You'll see the cached objects and queries in this window.

3. We'll have to click on **Show Info** from the **Action** menu list. This will open the cache statistics.

4. Now you'll see the **Cache information** window and all the statistics regarding cache.

How it works...

Whenever queries are executed, these statistics will be refreshed automatically. So you can easily monitor the cache hits from this tool.

Managing the cache

Managing the cache is very critical in the BI Server. Although it brings many advantages, if it's not managed successfully end users may access old results from the cache.

How to do it...

1. We need to purge the cache whenever the data warehouse is refreshed or updated. We can achieve this by defining the **Cache Persistence Time** value. This setting is on the **General** tab of the Physical table. If you set it as one day, then the lifetime of this table in the cache will be a maximum of one day.

2. The second method is more popular. You'll need to create a job that will execute the script as follows:

   ```
   SAPurgeAllCache()
   ```

   When you run this script, either by using **Job Manager** or manually from Presentation Services, the entire cache will be cleaned up and all the cache entries will be deleted. To open **Job Manager**, click on the **Manage** menu list and select the **Jobs** menu item.

3. Actually, purging the cache is an important task but also seeding the cache is another important task. To seed the cache you'll have to schedule the analyses to be executed according to the schedule. Scheduling of the analyses is going to be made by creating an Agent.

How it works...

Let's assume that the data warehouse is being refreshed every midnight. Then we'll have to refresh the cache right after the data warehouse is updated. First of all, we're going to purge the cache. Then the next step will be seeding the cache by creating Agents. You are going to select dashboard pages or analyses in the agent configuration to run these reports during non-business hours. This task will seed the cache.

7

Creating Simple Oracle Business Intelligence Analyses

In this chapter, we will cover:

- ▶ Constructing the analysis
- ▶ Exploring the table view properties
- ▶ Formatting the table view
- ▶ Filter types and creating the filters
- ▶ Using the selections
- ▶ Adding column prompts

Introduction

In this chapter, we're going to cover how to construct the analysis in Presentation Services. We've already discussed the back-end side of the BI Server. Now we're going to learn how to use the presentation tables that are stored in the BI repository.

After building the BI repository, it's time to execute queries against the organized data by using a set of graphical tools. OBIEE enables end users to construct web-based reports easily. We don't have to install anything on the computers to construct analysis. We're just going to log in to Presentation Services and access the Analysis Editor.

So we're going to create a simple analysis and then explore the default properties of the analysis. Formatting techniques are going to be covered in this chapter and we'll also learn how to eliminate some rows from the result set by using filters, selections, and column prompts. This will allow end users to create their own reports and focus only on certain business data.

Constructing the analysis

The Analysis Editor allows users to create and modify the analyses. These analyses are going to be the building blocks of the dashboards. The analyses can be saved into the Presentation Catalog and can be published in the dashboards. They are reusable presentation objects.

We're going to learn how to construct the analyses in this recipe. Also some of the advanced features are going to be covered in the next chapter.

How to do it...

1. When we first access Presentation Services, you'll see **Create** section. Clicking on **Analysis** in the **Create** section will pop up the **Select Subject Area** pane. You're going to select one of the existing subject areas that you're interested in. Also, you can click on the **New** button on the toolbar to access the **Analysis** link. So you can access Analysis Editor in both ways.

2. After you click on the **Sales** subject area, Analysis Editor is going to be opened as it's displayed in the following screenshot. You'll see the **Subject Areas** pane on the left-hand side in the **Criteria** tab. It contains the list of the presentation tables that are already defined in the Oracle BI repository. You'll find the **Selected Columns** pane next to the **Subject Areas** pane. You'll drag-and-drop the presentation columns onto this pane. There are also **Catalog** and **Filter** panes below the screen. The **Catalog** pane displays the presentation objects that you can use in this analysis. The **Filters** pane shows the existing filters, if there are any.

3. For this demonstration, we'll use two dimension columns and one measure column. We can easily drag-and-drop these three columns from the **Subject Areas** pane to the **Selected Columns** pane one-by-one.

   ❑ The `Region` column

   ❑ The `State` column

❑ The `Dollars` column

4. After selecting columns, we're going to click on the **Results** tab. The query is going to be executed and it'll display the result set. It creates one **Compound Layout** for this analysis and adds the **Title** and the **Table** view automatically.

5. Now, we're going to check the properties of the columns. There are different kinds of actions that we can perform. When you click on the button next to the column name, you'll see that the menu list is displayed. You can change the sort order of the data from this menu. If you have already sorted the data based on one column, you can still add secondary and subsequent sort order rules on the other columns. There are also many attributes that you can define in **Column Properties**.

6. After you click on the **Column Properties** option from the opened menu list, it'll display the **Style** tab in the **Column Properties** window. You can change the type, size, and color of the font in the **Font** section. Also the cell alignment rules are defined in this tab.

7. Then we click on the **Column Format** tab to change column properties. By default, column name is retrieved from the repository, but if you want to use a different name, then you'll have to select the **Custom Headings** checkbox and change the value of the **Column Heading** attribute. We can also hide this column from the analysis by selecting the **Hide** checkbox. If you want to display repeated values in the analysis, you can select the **Repeat** option box.

8. You can also change the data type format in the **Data Format** tab. By default, it retrieves the format settings from the repository. We can easily change this format type depending on the business requirements, by selecting the **Override Default Data Format** checkbox.

9. We can also define conditional formatting on a column to display the data in different colors based on their values. Conditional formatting will be defined in the **Conditional Format** tab. The condition of a column can be based on either the same column values or the other column values. When you click on the **Add Condition** button, you will see all the columns in the analysis.

10. You can also configure the interaction rules in the **Interaction** tab. There are two kinds of interactions. One is based on the column name and the other is based on the column values. Default interaction is set to **Default (Drill)**. Obviously, this drill interaction depends on the hierarchies. Even though the default interaction is drill, if there's no hierarchy, then this drill interactivity is going to be inactive.

11. By default, the write-back feature is disabled in the analysis. It requires additional settings in the repository so that this cannot be enabled in the tab right now. But if you want to allow end users to change the values in the analyses and store the new values in the database, you can enable the write-back feature. This is done by enabling the write-back feature in the repository and then selecting the **Enable Write Back** checkbox.

How it works...

When you create the analysis, it uses the default options. It creates one compound layout and adds the title and table views. So all the additional formatting options should be defined by the end users. These options will be saved in the analysis so wherever they're published, these options will also be available.

The formatting options will be applied during runtime. When you execute a query, the result set will be retrieved from the database and then all these options will be applied.

There's more...

The options we covered in this recipe are all related with the column properties. These options can be copied and used in another analysis easily. So you don't have to perform the same tasks on the other analysis. It's recommended to create some analyses just to store these formatting options and we can reuse them when needed.

Exploring the table view properties

We already noticed that the **Title** and **Table** views are added to **Compound Layout** when the Analysis is first created. According to business requirements, we may need to modify the **Table** view properties. For example, Subtotals and Grand Totals are not displayed in the default layout. Some additional modifications can be performed in the **Table** views:

▶ Table prompts
▶ Sections
▶ Subtotals and Grand Totals
▶ Excluded columns

How to do it...

1. We're going to create a simple analysis and click on the **Results** tab in the Analysis Editor. You will see that the **Title** and the **Table** views are added by default. You'll need to click on the Pencil icon to open the table view editor.

2. When the table view editor is opened, you'll see the **Layout** section below the **Results** section. The results will be automatically previewed according to the settings that you've changed. The **Layout** section is divided into four parts:

   ❑ **Table Prompts**

   ❑ **Sections**

❑ **Table** - **Columns and Measures**

❑ **Excluded**

3. We're going to drag the `Region` column on to the **Table Prompts** section in **Layout**. You'll notice that the preview will be refreshed and a prompt will appear in the result section. It will show the prompt as a drop-down list and distinct `Region` column values will appear in this list.

4. Also Subtotals and Grand Totals can be added to the analysis. You'll need to click the Total icon in the corresponding section. To close the table view editor, click on the **Done** button.

5. After the totals are added to the analysis, they will appear in the analysis. In this example, we have already added the subtotals for the `Region` column.

6. If you want to exclude any column from this **Table** view, you'll just need to drag the column onto the **Excluded** section. You'll see that the column won't be displayed in the **Table** view. In this example, the State column is excluded.

7. You'll also see the table view properties button when the table view editor is opened. Clicking on the properties button will pop up the **Table Properties** window. In the **Style** tab you can find the formatting options regarding that table view.

8. When you click on the **Enable Write Back** tab in the **Table Properties** window, you will see the write-back features of this view.

How it works...

The settings that we've made using the table view editor will be specifically related only to that table view. It won't be a reusable object. All these settings will be a part of the analysis definition. So they will be saved into the Presentation Catalog when you save the analysis.

Formatting the table view

Business users will need to focus on crucial data in the analysis, so displaying formatted data is very important. One analysis may consist of hundreds of rows but users should be able to focus only on important ones from the point of business view. In this section, we're going to enable **conditional formatting** in the analysis. We'll use these columns in the sample analysis:

- ▶ The `Region` attribute column
- ▶ The `State` attribute column
- ▶ The `Dollars` measure column

We are going to enable conditional formatting on the `Dollars` column. We'll also change the sort order of the result set.

How to do it...

1. We're going to create a sample analysis with the mentioned columns. Then we'll need to open the **Column Properties** of the `Dollars` column. We'll click on the **Add Condition** button in the **Conditional Format** tab. All available columns will be displayed. The condition of the `Dollars` column values may be dependent on another column's value. But we're going to use the `Dollars` column's values to format the values. So select the `Dollars` column.

2. The **New Condition** window will pop up and we'll define a condition. There are many operators that we can use for this condition. For this example, we're going to use the **is less than** operator. We can set any value in the **Value** textbox according to business requirements. We'll set it as `15000000`. Then we'll click on the **OK** button.

3. Right after we click on the **OK** button, the **Edit Format** window will pop up. We can define the **Font**, **Cell**, or **Border** formatting options in this window. We'll only change **Background Color** to red. So if the `Dollars` column's value is less than 15000000, then it'll be displayed with a red background color. Then click on the **OK** button.

4. We're going to add the next two conditions as the first one. After adding the other two conditions click on the OK button. According to our scenario if the `Dollars` column's value is:

   ❑ Less than 15000000, then show it with red color

   ❑ Between 15000001 and 25000000, then show it with yellow color

❑ Greater than 25000000, then show it with green color

5. Business users will need to see the currency symbol in the analysis. So we're going to change the data format in the **Data Format** tab of the `Dollars` column. Select the **Override Default Data Format** checkbox and select the **Currency** option from the **Treat Number As** drop-down list. Then click on the **OK** button to close the **Column Properties** window.

6. We're going to change the sort order of the result set based on the `Dollars` column values. Click the More Options icon next to the `Dollars` column to see the menu List. Then click on the **Sort** menu item and select the **Sort Descending** option. After configuring the sort order rule, click on the **Results** tab to see what the analysis looks like.

How it works...

You'll see that the result set is ordered by the `Dollars` column values in descending order. Also, the `Dollars` column values are conditionally formatted. Their background colors are formatted based on the values.

There's more...

Conditional formatting enables users to focus on crucial business data. But the conditions that we used in this scenario are based on static values. So they should be maintained as time passes. This will increase the maintenance cost. It's better to use variables to make them dynamic values. Another solution is using the **Key Performance Indicators** (**KPI**). We're going to cover the KPI definitions in *Chapter 9*, *Measuring Performance with Key Performance Indicators*.

Filter types and creating the filters

When we create any analysis in OBI, the query will be generated by the BI Server and executed against the database. Regarding the selected columns, all rows will be retrieved. End users should only see the data that they are interested in. So we're going to create the filters in the analyses to eliminate some rows from the result set.

We're going to create another sample analysis that includes the following columns:

- ▸ The `Region` attribute column
- ▸ The `State` attribute column
- ▸ The `Units Ordered` measure column
- ▸ The `Units Shipped` measure column

How to do it...

1. We're going to click on the more options icon next to the `Region` column to access the menu list in the **Criteria** tab. Select the **Filter** menu item from the list.

2. The **New Filter** window will pop up and we're going to select one of the operators from the **Operator** drop-down list. In this scenario, we're going to use the **is equal to / is in** operator in the filter. When you click the **Value** textbox, it will show the available values. Select the **East** value and click on the **OK** button to close the **New Filter** window.

3. Then click on the **Results** tab to see the result set that consists of only the `East Region` values.

4. Instead of selecting the values, we can also select other options by clicking on the **Add More Options** button in the **Edit Filter** window. The **Edit Filter** window is going to pop up when you edit the existing filter in the **Criteria** tab. You can use an SQL Expression or an existing variable. But unfortunately, the variables won't be displayed automatically. You'll need to write the variable name manually. Besides these options, you may be interested in calling the database functions. There are three ways to call the database functions directly:

    ❑ The `EVALUATE` function: Used for scalar and analytic calculations

    Syntax: `EVALUATE('DB_Function(%1)',Expressions)`

    ❑ The `EVALUATE_AGG` function: Used for aggregate functions with **group by** clauses

    Syntax: `EVALUATE_AGGR('DB_Aggregate_Function(%1)', Expressions)`

    ❑ The `EVALUATE_PREDICATE` function: Used for functions with a return type of Boolean

    Syntax: `EVALUATE_PREDICATE('DB_Function(%1)', Expressions)`

5. If you have already saved a filter, you can also use this filter in any analyses. You can see the list of saved filters from the **Catalog** pane. We'll drag-and-drop the filter from the **Catalog** pane to the **Filters** pane in the **Criteria** tab.

6. When you drag-and-drop the saved filter, `RegionWestFilter`, the **Apply Saved Filter** window will pop up so that we can configure the settings of the saved filter. There are two checkboxes that are specified below. Without selecting any checkbox, just click on the **OK** button.

    ❑   **Clear all existing filters before applying**: By selecting this checkbox, all existing filters will be deleted from the analysis.

❑ **Apply contents of filter instead of a reference to the filter**: If you don't select this checkbox, there will be a link between the analysis and the saved filter. If you change the filter content in a future time, the analysis is also going to be affected. But if you select this checkbox, the definition of the filter will be copied to the analysis and there won't be any reference to the saved filter.

How it works...

After adding the saved filter, click on the **Results** tab and check the result. You'll only see the `West` values in the `Region` column.

This time, the BI Server is going to generate the SQL query with a `WHERE` criteria. So the data that will be retrieved from the database will be eliminated based on this criteria.

These are the examples of filter definitions. These filter definitions are not going to be changed by the end users once these analyses are published in the dashboards. Sometimes we'll need to create some analyses that will allow end users to interact by selecting a value from a drop-down list. These settings will be implemented by adding **Column Prompts** in the analyses at the end of the chapter.

Using the selections

When the analysis is constructed, it'll show all the data as it's in the database. We can implement some filters or grouping options in the analysis after the data is aggregated.

In order to make the demonstration, we're going to use a new analysis that consists of these columns. You can see this in the following screenshot.

- ▶ The `Customer` hierarchy column
- ▶ The `Dollars` measure column

How to do it...

1. After constructing the sample analysis, click on the **Results** tab. We'll see that this time, **Compound Layout** consists of the **Title** and **Pivot Table** views. This happened because we have selected a hierarchy column instead of selecting an attribute column.

2. When you expand the **Selection Steps** pane in the **Results** tab, you'll see that the **ALL** members option will be displayed by default. Click on the **Then, New Step** link and navigate to **Add Groups or Calculated Items | New Group**.

3. The **New Group** window will pop up. You can select the members to create a new group. We're going to select the `West` and the `East` values. You can use `West & East` as the group name. Then click on the **OK** button.

How it works...

When you see the result set, you'll notice that a new member is added as a group into the **Pivot Table** view. It's called as `West & East` and it shows the total `Dollars` value based on the two regions.

These selection steps are very useful when you want to display comparable report values in the analyses.

There's more...

Instead of adding a new group, we could also remove an existing member from the **Pivot Table** view. Both are supported. For example, if we had removed the West value from the **Selection Steps** pane, it would only show the `Central` and the `East` values.

Adding column prompts

When it comes filtering data dynamically, it'll be good to allow end users to select the values from a drop-down list. We can achieve this by using **Column Prompts**. To demonstrate the column prompts, we're going to use a new analysis that consists of four columns as you'll see in the following screenshot:

How to do it...

1. After creating the sample analysis, click on the **Prompts** tab. Click on the Add Prompt icon and go to **Column Prompt | "Dim_Customer"."Region"**.

2. The **New Prompt: Region** window will pop up on the screen. Set the **Label** value as `Choose a Region` and don't change any settings in the windows. Just click on the **OK** button.

3. You'll see that the new column prompt is created. Create one more column prompt that's based on the `State` column this time. But in this one, we're going to select the **Limit Values by** checkbox and select the **Choose a Region** column prompt. This setting will make the `State` values be populated based on the `Region` column values. Click on the **OK** button again.

4. As a result you'll see two column prompts in the **Prompts** tab.

How it works...

1. When an analysis has a column prompt in its definition, it doesn't show the result set immediately. Instead, it shows the column prompts to the end users. The users should select the values from the drop-down lists.

2. After selecting the values, they should click on the **OK** button in order to see the result set based on their selections.

8

Adding Views
to Analyses and
Advanced Features

In this chapter, we will cover:

- ▶ Adding the pivot table view
- ▶ Adding the graph view
- ▶ Adding the gauge view
- ▶ Adding the legend view
- ▶ Adding the column selector view
- ▶ Adding the view selector view
- ▶ Configuring the master-detail view settings

Introduction

End users will like to see different kinds of reports in the dashboards. They will be interested in interactions on the reports. Table and title views may not satisfy their business requirements. So we're going to see how to add different views into the compound layouts in this chapter.

We're going to add the pivot table view, graph view, gauge view, and so on into the analyses to improve the functionality of the reports. You'll also find the customization steps of these new views.

One of the most important features that we're going to cover is the master-detail view setting. After setting up the master-detail view configuration, end users will be able to see the interaction between the two views inside the analysis. This feature will let BI developers create small number of analyses, so the maintenance cost will be also reduced.

Adding the pivot table view

We're going to add a pivot table view into the analysis. Normally, when you add an attribute and a measure column in the **Criteria** tab, the title and table views are added to the compound layout of the analysis. But if you've already created a hierarchy column in the repository, you have an option to select that column. If a hierarchy column is added, then the compound layout will consist of the title and pivot table views.

How to do it...

1. We're going to add these columns into the analysis in the Criteria tab as shown in the following screenshot:

   ❑ `Product`: Hierarchy column

   ❑ `YEAR`: Attribute column

   ❑ `Dollars`: Measure

2. When you click on the **Results** tab, you'll see that two views are automatically added to the **Compound Layout**. As usual the **Title** view is added. You'll also find the **Pivot Table** view.

3. We need to click on the Edit View (pencil sign) button, to open the pivot table view editor so that we can customize the default pivot table view content. By default, the `Product` and `YEAR` columns are added to the **Rows** section and the `Dollars` measure column can be found in the **Measures** section.

4. We're going to drag-and-drop the `YEAR` column from the **Rows** section to the **Pivot Table** prompts area to have a drop-down list that will show the distinct values of the `YEAR` column.

5. Another option is to show the `YEAR` column values in the **Columns** section below the `Measure Labels` column. You can simply drag-and-drop the `YEAR` column into the **Columns** section.

6. After adding the YEAR column into the section, we'll configure the pivot table view to show the Grand Total for the rows. Click the Totals After button and select the **After** option from the menu list.

7. You can also add a graphic next to the pivot table. Click on the Graph Pivoted Results button in the view editor.

8. When you expand the `Product Total` column, you'll see that the graphics will be automatically refreshed and it will display the bar graphic based on type code column values.

How it works...

We've created an analysis that consists of two views. As usual the first one is the title view and the second one is the pivot table view. If you use attribute columns when you're constructing the analysis, table view is automatically added to the view. But if you've used hierarchy columns in the **Criteria** tab, then the pivot table view will be added instead of the table view. In our scenario we have used hierarchy columns. One of the important features of the pivot table views is that we can easily navigate to finer levels by expanding the hierarchy. Pivot table views allow end users to access the finer levels in the same view without navigating to any other page. Instead of drilling down, you expand the members in the hierarchy.

We can also add the attribute columns into the **Columns** sections instead of the **Rows** section to have a different way of presenting the reports.

Adding the graph view

Although table and pivot table views are very useful views in the analyses, they might not satisfy all the business requirements. Business users may be interested in some visual effects in the reports. To support this requirement, we're going to add graph views into the compound layout. This will enable end users to focus on the business data easily. There are also other types of graphical views and we're going to discuss them after this recipe.

How to do it...

1. We're going to demonstrate this feature with a two-column analysis. So add one attribute and one measure column in the **Criteria** tab.

   ❑ `Region`: Attribute column

   ❑ `Dollars`: Measure column

2. To see the result set, just click on the **Results** tab. As usual, the **Title** and the **Table** views are added to **Compound Layout**. Click on the New View button and go to **Graph | Bar | Default (Vertical)** from the menu list.

3. You'll see that the **Graph** view is added into the **Compound Layout** section below the table view.

4. We have to open the view editor in order to customize the default graphic. Once the graph view editor is opened, you'll see similar properties that we've already discussed in the pivot table view. Now we're going to learn how to implement conditional formatting in the graph view. Clicking on the Graph View Properties button in the view editor will cause the **Graph Properties** window to pop up. After accessing the **Style** tab, click on the **Style and Conditional Formatting** button.

5. The **Style and Conditional Formatting** window will pop up. There are two tabs named **Style Formatting** and **Conditional Formatting**. Click on the **Conditional Formatting** tab, and then click on the **Add Condition Format** button and select the `Dollars` column in the menu list.

6. You're going to select the **is less than** operator in the **New Condition** window and set a value of say `15000000`. Also, changing the background color of the cell will enable end users to focus on the important data. You can add multiple conditions but we're going to use only one condition in our sample scenario.

7. When you're done in the view editor, you'll see that the bar graphic is conditionally formatted.

How it works...

Different graph view types can be used in the analyses and you can also add multiple graph views into the existing analysis. The default **Bar** type is **Vertical Bar**. You have other options in bar graphics such as **Horizontal Bar**, **Stacked Vertical Bar**, and **Stacked Horizontal Bar**.

You may also use different graphic options other than **Bar** such as **Line Graphics**, **Area Graphics**, **Pie Graphics**, and so on. The complete list is available when you click on the New View button and select the **Graph** menu option.

You've already seen how to add a graph view into an existing analysis that consists of title and table views. When you use table and graph views in the same compound layout, there will be an interaction between these two views. If the Drill Down feature is enabled on the column values in the table view, then end users may drill down to the finer levels. At that time, graph view is going to be refreshed automatically and that finer level data will be displayed in the graphic.

Adding the gauge view

Another type of view is the gauge view, which can be easily added to the analysis. This type of view is also very useful to improve the visual appearance in the dashboards. They display the results in gauges such as **Dial**, **Bar**, and **Bulb** style gauges. They are very useful in displaying performance against the goals and gauge views are very effective in displaying single data value.

How to do it...

1. We're going to create another analysis that consists of two columns. After displaying the result, click on the New View button and go to **Gauge | Default (Dial)**.

   ❑ `Region`: Attribute column

   ❑ `Dollars`: Measure column

2. The dial type of the **Gauge** view will be added below the **Table** view.

3. To customize the gauge view, click on the Edit View button of the **Gauge** view. This will pop up the view editor. You can easily change the size of the gauges that are created based on the `Region` column values. Select the **Medium** option from the drop-down list to resize the gauges.

4. The threshold values are calculated automatically but if you want to specify these values manually, you can easily specify values in the **Settings** section by clicking on the Threshold values button. You'll have some options regarding threshold values:

   ❑ **Dynamic**: This is the default option

   ❑ **Custom Value**: You can set the value explicitly

   ❑ **Variable Expression**: You can use presentation or repository variables

   ❑ **SQL Query**: An SQL query may populate the threshold values automatically during runtime

5. Clicking on the gauge view properties button will pop up the **Gauge Properties** window. You can set the property values specified as follows:

   ❑ **General**

   ❑ **Style**

   ❑ **Scale**

   ❑ **Titles and Labels**

How it works...

Having many gauges in the analysis may cause loss of focus. You'll need to set the number of gauges carefully. You can achieve this by using gauge prompts to have a drop-down list, so that end users may select the value that they're interested in. Also you can easily change the gauge type. Gauge views have three types, other than the default type:

- **Dial**: Default option
- **Vertical Bar**
- **Horizontal Bar**
- **Bulb**

Adding the legend view

When you use the gauge and graph views in the analyses, you'll also have to inform the end users about the meaning of the colors that are used in these views. Otherwise it may cause confusion. The technical solution for this business challenge is using the legend views.

How to do it...

1. We're going to use an analysis that consists of three views in this demonstration, title, table, and graph views. As you can see in the screenshot, conditional formatting is implemented in the graph view. We're going to click on the New View button and go to **Other Views | Legend**.

2. The **Legend** view is going to be added below the **Table** view. It doesn't display any information by default. So to edit the content of the legend, click on the Edit View button.

3. When the view editor is displayed, just click on the **Add Caption** button so that there will be two captions. In our scenario, we need only two because we've used only two colors in the graph view. Also enter a value in the **Title** textbox.

4. Set description values in the **Caption** textboxes and then you'll need to click on the Format Text button to select the proper color.

5. The **Format Text** window will pop up so that you can make necessary changes to format the legend data. We're going to only set the background color in this scenario.

How it works...

After customizing the legend content, you'll see the **Legend** view below the **Table** view. You can also change the order of these views by just a drag-and-drop.

Adding the column selector view

When you start publishing the analyses in the dashboards, you'll notice that there won't be so much free space in the dashboards. While some users are interested in a report that consists of the `Type Code` and the `Dollars` columns, maybe others will be looking forward to seeing a similar report that consists of the `Year` and the `Dollars` columns. You won't be able to create many analyses and display them in the dashboard.

So we're going to use the column selector view for this business challenge.

How to do it...

1. Create an analysis that consists of the `Type Code` and the `Dollars` column. Then click on the New View button and navigate to **Other Views | Column Selector**.

2. The **Column Selector** will be added below the **Table** view. Click on the Edit View button to open the view editor. Select the **Include Selector** checkbox for Column 1, that is `Type Code`.

3. Then double-click on the `YEAR` column on the subject areas pane. This will add the `YEAR` column to the **Column 1** part. Repeat the same task for the `Region` column in the `Dim_Customer` presentation table.

4. Close the View Editor. You'll see that **Column Selector** will appear below the **Table** view. Drag the **Column Selector** and drop it between the **Title** and the **Table** views. It displays the default column in the drop-down list.

How it works...

Now the analysis displays `Type Code` column values in the table. When you select the `YEAR` column from the drop-down list, the table view will get automatically refreshed and it'll display the `YEAR` column.

Or you can also select the `Region` value from the drop-down list to display the `Region` and the `Dollars` columns in the analysis.

Adding the view selector view

You can add new views; as many as you want. But again, when it comes to publishing these views in the dashboards, you'll have some difficulties with fitting them in the dashboards. You can easily use the view selector view to enable end users to select one of the existing views.

How to do it...

1. In this scenario, we're going to use an analysis that consists of four views:

   ❑ Table view

   ❑ Graph view

   ❑ Legend view

   ❑ Gauge view

2. In the **Results** tab, click on the New View button and navigate to **Other Views** | **View Selector**.

3. Once **View Selector** is added, click on the Edit View button to open the view editor. Then enter `Select a View` in the **Caption** textbox. Select all the existing views in the **Available Views** list and click on the right arrow button. So all the views will be displayed in the **Views Included** list.

4. Close the view editor and remove all the views from the **Compound Layout** by clicking on the Remove View from Compound Layout button, except for the **View Selector** view.

How it works...

The only view that will be displayed in the **Compound Layout** is the **View Selector** view. So end users may select the views that they're interested in. Select the **Table** value from the drop-down list and you'll notice that a table will be displayed.

Then select **Graph** from the drop-down list so that a graph alone will be displayed.

Configuring the master-detail view settings

Business users may be interested in interaction between two views in the analysis. Whenever they click on a value in the table view, the view below it will be refreshed automatically and the data that the user wants to focus on will be displayed. This can be achieved by configuring the master-detail view settings.

How to do it...

1. In this example, we're going to use an analysis that consists of the table and graph views. Graph view displays the `Region` and the `State` values.

2. Open the graph view editor to make necessary changes in the detail view. First drag the `Region` column to the **Sections** area and then select the **Display as Slider** checkbox. At that moment, a slider will appear above the graphic that displays the `Region` column values.

3. Clicking on the graph properties button will pop up the **Graph Properties** window. Select the **Listen to Master-Detail Events** checkbox and set a value for the **Event Channels** textbox, `ChRegion`.

4. Close the graph view editor and check the analysis in the **Results** tab. You'll see that a slider appears above the graphic. The **Table** view also shows the list of regions. But by default, the interaction of the column values is set to drill down. If you click on a value in the `Region` column it's going to drill down to a finer level.

5. Now we're going to change this default interaction type. Go to the **Criteria** tab and click on the More Options button. Then select the **Column Properties** option in the `Region` column.

6. Change the **Primary Interaction** value to the **Send Master-Detail Events** and enter a value for the **Specify channel** textbox, `ChRegion`. (This value should be exactly same as the one you've already entered in the graph view properties.)

How it works...

To test this configuration, go to the **Results** tab. You will see that the `Region` column values are not displayed as hyperlinks. After clicking on the `West` value, you'll see that the value in the slider will focus on the `West` value and the **Graph** view will only display the state values from the `West` region.

9
Measuring Performance with Key Performance Indicators

In this chapter, we will cover:

- ▶ Creating the KPIs and the KPI watchlists
- ▶ Creating the scorecards
- ▶ Creating the objectives
- ▶ Creating the initiatives
- ▶ Adding the perspectives
- ▶ Building the strategy trees and maps

Introduction

Publishing the analytical reports in the dashboards will not satisfy the business requirements always. Business users will be interested in some reports so that they can compare their values. You also need to display the reports that will enable users to focus on only the most critical business entities. Sometimes this can be achieved by using the conditional formatting feature. But the defined threshold values are going to be static values. As time passes, you'll need to reset these values, so it has a maintenance cost. To solve these business challenges, you can use Key Performance Indicators (KPI). They can be used to measure the productivity of an organization relative to its objectives.

KPIs are widely used in OBIEE 11*g* and you can easily publish these KPIs in the dashboards by creating KPI watchlists.

Key Performance Indicators are also building blocks of strategy management. In order to implement balanced scorecard management technique in an organization, you'll first need to create the KPI objects. Then you're going to create the scorecards and use the KPIs.

We're going to cover KPIs, KPI watchlists, and also scorecards in this chapter. You'll also learn how to create strategy trees and maps at the end of this chapter. After implementing these features, you're going to easily measure the performance of the organization.

Creating the KPIs and the KPI watchlists

We're going to create Key Performance Indicators and watchlists in the first recipe. There should be comparable measure columns in the repository in order to create KPI objects. The following columns will be used in the sample scenario:

▸   `Shipped Quantity`
▸   `Requested Quantity`

How to do it...

1.   Click on the **KPI** link in the **Performance Management** section and you're going to select a subject area.

The KPI creation wizard has five different steps.

2. The first step is the **General Properties** section and we're going to write a description for the KPI object. The **Actual Value** and the **Target Value** attributes display the columns that we'll use in this scenario. The columns should be selected manually. The **Enable Trending** checkbox is not selected by default. When you select the checkbox, trending options will appear on the screen. We're going to select the `Day` level from the `Time` hierarchy for trending in the **Compare to Prior** textbox and define a value for the **Tolerance** attribute. We're going to use `1` and **% Change** in this scenario.

3. Clicking on the **Next** button will display the second step named **Dimensionality**. Click on the **Add** button to select `Dimension` attributes.

4. Select the `Region` column in the **Add New Dimension** window.

5. After adding the `Region` column, repeat the step for the `YEAR` column. You shouldn't select any value to pin. Both column values will be **<is prompted>**. Clicking on **Next** button will display the third step named **States**.

Dimensionality
Define dimension values for the KPI or allow each dimension to be prompted by the user.

Define KPI value for

Dimension	Value
"Dim_Customer"."Region"	<is prompted> ▾
"Dim_Time"."YEAR"	<is prompted> ▾

Add...   Remove

6. You can easily configure the state values in this step. Select the **High Values are Desirable** value from the **Goal** drop-down list. By default, there are three steps:

    ❑ **OK**

    ❑ **Warning**

    ❑ **Critical**

States
KPI state is determined by comparing the actual (**"Fact_Sales"."Shipped Quantity"**) and target (**"Fact_Sales"."Requested Quantity"**) values. For each KPI state, define a label, color, and icon. Determine state ranges by defining threshold values between each state.

State Properties

Goal High Values are Desirable ▾
High Values are Desirable
Low Values are Desirable
Target Value is Desirable

	Color	Icon	Actions		
OK	▾	✓	✋	*greater than 100%*	Thresholds ☑ define as % of target value
Warning	▾	⚠	✋	*between 90% and 100%*	100 ▲▼ %
Critical	▾	✖	✋	*less than 90%*	90 ▲▼ %

If KPI returns No Data No status ▾

7. Then click on the **Next** button and you'll see the **Related Documents** step. This is a list of supporting documents and links regarding to the Key Performance Indicator.

8. Click on the **Add** button to select one of the options. If you want to use another analysis as a supporting document, select the **Catalog** option and choose the analysis that consists of some valuable information about the report.

9. We're going to add a link. You can also easily define the address of the link. We'll use the `http://www.abc.com/portal` link.

10. Click on the **Next** button to display the **Custom Attributes** column values. To add a custom attribute that will be displayed in the KPI object, click on the **Add** button and define the values specified as follows:

   ❑ `Number: 1`

   ❑ `Label: Dollars`

   ❑ `Formula: "Fact_Sales"."Dollars"`

11. Save the KPI object by clicking on the **Save** button.

Right after saving the KPI object, you'll see the KPI content.

12. KPI objects cannot be published in the dashboards directly. We need KPI watchlists to publish them in the dashboards. Click on the **KPI Watchlist** link in the **Performance Management** section to create one.

13. The **New KPI Watchlist** page will be displayed without any KPI objects. Drag-and-drop the KPI object that was previously created from the **Catalog** pane onto the KPI watchlist list.

14. When you drop the KPI object, the **Add KPI** window will pop up automatically. You can select one of the available values for the dimensions. We're going to select the **Use Point-of-View** option. Enter a **Label** value, `A Sample KPI`, for this example.

15. You'll see the dimension attributes in the Point-of-View bar. You can easily select the values from the drop-down lists to have different perspectives. Save the KPI watchlist object.

How it works...

KPI watchlists can contain multiple KPI objects based on business requirements. These container objects can be published in the dashboards so that end users will access the content of the KPI objects through the watchlists.

When you want to publish these watchlists, you'll need to select a value for the dimension attributes.

There's more...

The Drill Down feature is also enabled in the KPI objects. If you want to access finer levels, you can just click on the hyperlink of the value you are interested in and a detailed level is going to be displayed automatically.

Creating the scorecards

Scorecard is a strategy management tool that will help you to measure the productivity of the organization. It has four different components to build up a scorecard:

- ▶ **Strategy**: This is the objective of the organization. There can be multiple objectives in the same organization depending on the divisions.

- ▸ **Initiatives**: These are the time-specific tasks or projects. You'll use initiatives to support the objectives.

- ▸ **Scorecard Documents**: These documents will contain supporting information about the strategy management.

- ▸ **Perspectives**: These are different point of views of the organization. There are four different types of perspectives by default. You can add more depending on the organization requirements. These four perspectives were proposed by Robert S. Kaplan and David P. Norton.

How to do it...

1. Click on the **Scorecard** link in the **Performance Management** section.

2. The **New Scorecard** window pops up. Enter a name for the scorecard. If the **Use Default Perspectives** checkbox is selected, default perspectives will be loaded into this scorecard. Then click on the **OK** button.

3. The scorecard editor will pop up in the screen. Only the default perspectives will be loaded into this scorecard. Save this scorecard object.

How it works...

Scorecards consist of objectives, initiatives, documents, perspectives, and KPIs. Key Performance Indicators and supported documents should be designed and created before you start working on scorecards. The rest of the definitions are the features of the scorecard.

Creating the objectives

The performance management strategy of an organization is broken into objectives. Objectives of an organization should be defined in the scorecards. You can create multiple objectives that can be defined depending on business requirements.

How to do it...

1. Click on the Create Objective icon to define the objectives.

2. The **New Objective** tab will pop up, so we're going to define the properties of the objective starting with the name. Enter the name of the objective.

Enter the name of the objective as `Improve the Profitability`. We're also going to change the **Perspective** to **Financial**.

3. Now we're going to define the measure in the scorecard. Drag-and-drop the KPI object from the **Catalog** pane to the **Objectives & KPIs** list.

4. The new objective will look like the following screenshot:

How it works...

You can create multiple objectives in the scorecard. Every objective will focus on a particular perspective. Here's the sample scorecard that consists of multiple objectives.

Creating the initiatives

In this strategy management framework, initiatives are time-based action projects that are needed to support the organization to be successful in its strategy. You can create many initiatives to achieve the objectives. Initiatives are tied to the objectives and have significant importance on being successful in the organization's goals.

How to do it...

1. Click on the Create Initiative button to create it from the beginning.

2. The **New Initiative** page will pop up. Enter the name of the initiative. In our scenario, it will be `Upgrade the database`. You can set the **Start Date** and **Due Date** and you can also define the actions to achieve this initiative successfully.

How it works...

These time-based tasks will be helpful to achieve the objectives and the goals of an organization. One initiative may support multiple objectives.

There's more...

The progress of the initiatives can also be monitored and measured. The measuring will be dependent on KPI objects.

Adding the perspectives

Different divisions can concentrate on different point of views of the organization. There are four default perspectives as originally proposed by Kaplan and Norton. You can also define a different perspective to measure the productivity.

> ▸ **Financial**: This perspective is used to measure financial effectiveness of an organization

- ▸ **Customer**: This perspective focuses on the customer and it's used to measure customer satisfaction

- ▸ **Internal Process**: It refers to the quality of the internal processes

- ▸ **Learning and Growth**: This perspective is related to both individual and corporate self-improvement

How to do it...

Clicking on the New Perspective icon in the **Perspectives** pane will pop up the New Perspective tab. Then enter the name of the perspective.

How it works...

You can associate these perspectives with any objectives so they will be used with the organization's objectives. The list named **Objectives, Initiatives & KPIs** will display the dependent objects. You can easily find where they are used.

Building the strategy trees and maps

Strategy maps display the objectives and KPIs that have been defined in the scorecard. The parent-child objective hierarchy will also be displayed in the map. Strategy maps are very useful to see the progression in the organization.

How to do it...

1. Click on the View Strategy Tree icon in the **Strategy** pane, after selecting the objective or the scorecard.

The **Strategy tree** tab will be displayed. You can see the parent-child hierarchy in the screenshot.

2. Click on the **Details** tab and you will see detailed measures of the existing KPIs.

3. Go back to the **Diagram** tab and expand the KPI details to see the contents of the KPI and to access to the measure values.

How it works...

Strategy trees and maps will be very useful to see the bigger picture of the organization. End users can access these strategy trees and maps in the dashboards. This communication tool enables easy strategic communication.

10

Creating and Configuring Dashboards

In this chapter, we will cover:

- ▶ Creating the dashboards
- ▶ Using the Dashboard Builder
- ▶ Exploring the properties of dashboard objects
- ▶ Adding catalog objects to the dashboards
- ▶ Creating the dashboard prompts

Introduction

All the BI objects will be published in the dashboards. It's quite important to have well designed dashboards. We're going to discuss how to build and configure dashboards in this recipe.

End users are going access to dashboards to see the results of the analyses. Every user who is connected to Presentation Services can access their private dashboards and also we can create shared dashboards that will be common to a group of users.

Dashboards consist of the **dashboard pages**. When you create a dashboard, one dashboard page is created by default. For instance, we can create a dashboard named `Sales` and the dashboards pages can be created based on the location as follows:

- ▶ `New York Sales`
- ▶ `London Sales`
- ▶ `Paris Sales`

Also, interactivity in the dashboards will be needed. End users will be asked to prompt a value to change the content of the analyses that are already published. Dashboard prompts are also going to be covered.

Creating the dashboards

First step will be the creation of a new dashboard in Presentation Services. After creating a blank dashboard, we're going to configure its properties. Dashboards are stored as all the other catalog objects in Presentation Services. The difference is that they're the container objects. You can easily publish some other catalog objects in them.

How to do it...

1. To create a dashboard, click on the **Dashboard** link in the **Create...** pane.

2. The **New Dashboard** dialogue box will pop up. Enter the name and the description values as shown in the screenshot and then click on **OK** after selecting the **Add content later (Create empty dashboard)** option box.

   ❑ **Name**: `Sales`

   ❑ **Description**: `A Sample Dashboard`

After clicking on **OK**, it's going to display the home page of Presentation Services.

3. Let's create another dashboard but this time we are going to select the **Add content now** option box and click on **OK**.

4. Now, it's going to display the Dashboard Builder. The Dashboard Builder is used to modify the content of the dashboards. We're not going to add any catalog or dashboard objects now. Just close the Dashboard Builder and navigate to the home page.

How it works...

We have created two sample dashboards without any content. You can see that the new dashboards appear in the **Dashboards** menu. They are already saved as catalog objects in the Presentation Service's repository. Saving the dashboards into the shared folders enables them to be shared by all the users, depending on the permissions. Dashboards are also catalog objects. You can set permissions on them.

The `Sales` and `Financials` dashboards can be found in the catalog folders. These new dashboards are shared dashboards. End users are going to access them according to their privileges.

There's more...

These new dashboards are empty and don't contain any kind of catalog objects. We'll need to modify the content and also we are going to design the dashboards in the following section.

Using the Dashboard Builder

The Dashboard Builder is a tool that will be used to modify the dashboards. We're going to learn how to access the Dashboard Builder and also we're going to explore the properties of the dashboards.

How to do it...

1. Click on the `Sales` dashboard from the **Dashboards** menu to navigate to the `Sales` dashboard.

2. The `Sales` dashboard is going to be displayed on the screen and you'll see that there's no content in it.

3. There are two ways to open the Dashboard Builder. If the dashboard doesn't contain any object, you can click on the **Edit** link to navigate to the Dashboard Builder.

4. If there are objects already published in the dashboard then you can click and expand the properties menu and then select the **Edit Dashboard** link.

5. After using one of the methods, Dashboard Builder will be displayed. You will see that there are two panes in the left section:

    ❑ **Dashboard Objects**: These are the objects that can be used in the dashboards to extend the functionality of the dashboard.

    ❑ **Catalog**: These are the objects that we have created, such as analyses and KPIs. They are reusable objects.

6. There are also different buttons on the toolbar:

    ❑ 1 – This button adds a new dashboard page

    ❑ 2 – This button deletes the current page

❑ 3 – This icon opens the tool's list

❑ 4 – The **Preview** button gives you a preview of the dashboard page

❑ 5 – The **Run** button runs the project

❑ 6 – The Save icon will save the project

7. By default, there's only one page in the dashboard and it's called `Page 1`. Click on the **Add Dashboard Page** button to create a new page.

8. The **Add Dashboard Page** dialogue box will pop up. Write the name of the new page as `New York Sales`. Then click on the **OK** button. Repeat the same task for the following pages:

❑   London Sales

❑   Paris Sales

How it works...

We've created three additional dashboard pages so there are four pages right now, as displayed in the following screenshot. They're displayed as tabs. Whenever you want to modify one of the pages, you'll just need to click on the tab and then you can make any kind of modification.

Exploring the properties of dashboard objects

Now we're going to explore additional properties and features of dashboard objects to make customizations. End users may ask to change the order of the dashboard pages or they may be interested in changing the name of the pages. These customizations will be covered in this recipe.

How to do it...

1. Click on the Tools icon on the toolbar. There are five menu items in the list. Click on the **Dashboard Properties...** menu item in the list.

2. The **Dashboard Properties** window will pop up. Select the `Page 1` tab from the pages list.

3. Once the `Page 1` tab is highlighted in the list, click on the Rename button.

4. The **Rename** dialogue box will pop up. Write the new name of `Page 1` and click on the **OK** button.

5. You can also change the order of the pages in the dashboard by clicking on the Move selected items down button next to the pages list. Then click on **OK** to navigate to the Dashboard Builder page.

6. Click on the **Save** button to save the changes.

7. Then click on the **Run** button to navigate to the dashboard. This will cause the Dashboard Builder to be closed.

How it works...

Once the dashboard is saved, the Sales dashboard is going to be displayed. Now, you will see that there are four dashboard pages. All of the dashboard pages don't contain any kind of catalog object as yet.

Adding catalog objects to the dashboards

Now we're going to learn how to add objects into the dashboards. There are two types of objects that we can publish in the dashboards:

- ▸ Dashboard objects
- ▸ Catalog objects

Dashboard objects are used to publish additional information or actions in the dashboards. For example, if end users want to navigate to the company portal frequently, you can publish the company's web address in the dashboard. Or if an integration is needed between OBIEE and a third party application, we can create action links that will call a web service.

Catalog objects are the objects that are created by the BI developers, such as analyses and KPIs.

How to do it...

1. Navigate to the Dashboard Builder page and you will see that there's no content in the `Regional Sales` dashboard page.

2. Drag-and-drop one of the analyses from the **Catalog** pane to the dashboard content area. When you drop it, you'll see that two dashboard objects are automatically created. The **Column** and **Section** objects are automatically created because they are used to organize the web to publish the analyses.

3. Then drag-and-drop another analysis to the same section, below the first analysis. Now the `Regional Sales` dashboard page consists of two analyses.

4. Then simply drag the `Column`, `Section`, and `Folder` objects one by one next to the first `Column` object. Your work is going to look like the following screenshot. Then click in the Properties button of the folder object.

5. The **Folder Properties** dialogue box will pop up. Click on the **Browse** button to select a folder that will be published in the dashboard. Select the **Expand** checkbox and click on the OK button. Then save the dashboard.

How it works...

After saving and running the dashboard, your work will look like the following two screenshots. The dashboard page is divided into two columns. You'll see that the folder you have configured is displayed on the first column and it's expanded.

Two analyses are displayed in the second column. When the end users access the dashboard page, all of the analyses will be executed and the result set will be displayed.

There's more...

Design of the dashboards is very important. It shouldn't contain many analyses and other objects at same time. Users should be able to focus on the information easily. We're going to discuss the best practices of dashboard design in *Chapter 11, Oracle BI Best Practices*.

Creating the dashboard prompts

Dashboard prompts enable end users to interact with the analyses. Based on the values that are prompted, all the analyses are going to filter the results. These interactions are very useful and will be needed in the organizations frequently.

In order to gain the benefit of the dashboard prompts, we'll need to create analyses that have filters defined. Also, the filter criteria should match the criteria that were used in the dashboard prompts.

To demonstrate the dashboard prompts, we're going to create two analyses that consist of these columns:

- `Region`
- `State`
- `Dollars`

We're going to use the `Region` column as a filter. The analyses are as follows:

- The first analysis contains the **Region is prompted** filter

▶ The second analysis contains the **Region is equal to / is in East** filter

How to do it...

1. Click on the **Dashboard Prompt** link in the **Create** pane.

2. Click on the **Sales** subject area.

3. Click on the New button and select the **Column Prompt** option in the menu list.

4. The **Select Column** dialogue box will pop up. Select the `Region` column and click on the **OK** button.

5. Write the **Label** value in the **New Prompt: Region** dialogue box and deselect the **Enable user to select multiple values** checkbox. Click on the **OK** button and save the dashboard prompt in a shared folder.

6. Then add the dashboard prompt and the two analyses into a dashboard page. Your work will look like the following screenshot. Save the dashboard.

How it works...

1. The dashboard prompt and two analyses are published in the dashboard. When you first access the dashboard, a prompt will appear above the analyses. We've configured the dashboard prompt without a default value. So the analyses on the dashboard are not going to be affected at first.

2. But when the user selects a value from the dashboard prompt and clicks on the **Apply** button, the analyses will filter the data based on the selected value.

Region	State	Dollars
Central	IL	1,243,114
	IN	960,246
	LA	814,648
	MN	1,030,562
	MO	2,099,372
	NE	1,851,946
	OH	151,305
	OK	703,227
	TX	3,962,640
	WI	102,101

There's more...

If the defined filters in the analyses are set as protected, then the dashboard prompts will not affect the result set. In our scenario we have created the filters without the protected setting so that the dashboard prompt filters all the data in the analyses.

11
Oracle BI Best Practices

In this chapter, we will cover:

- ▶ Best practices of the Physical layer
- ▶ Best practices of the Business Model and Mapping layer
- ▶ Best practices of the Presentation layer
- ▶ Best practices of the analyses and the dashboards
- ▶ Performance and security tips

Introduction

The implementation process of Oracle Business Intelligence Enterprise Edition 11*g* should be very well planned and designed not to have a failed project. Every single step of this implementation is very crucial and any kind of mistake is going to impact the other dependent tasks. We should always focus on business requirements and also analyze technical features of the product in order to see the gap. We need to start with the business requirements and then we are going to drill down to the technical features. We're not going to discuss how to analyze the business needs in this book. Instead, technical details are going to be discussed.

In this chapter, we're going to cover the best practices of the OBIEE 11*g*. You will find useful recommendations about every layer of the implementation starting from the Physical layer of the repository to the dashboards.

Also you will find useful tips about performance and security in this chapter.

Before starting this recipe, we assume that the data warehouse is already designed and implemented.

Best practices of the Physical layer

You've already learned that building the Physical layer of the repository is the first step in OBIEE 11*g* implementation and hence all the initial steps are not going to be covered. You are going to find the best practices of the Physical layer in this recipe. We're going to discuss the common mistakes in the Physical layer and learn how to have a successful implementation of this layer.

How to do it...

1. Instead of creating the Physical layer objects manually, use the **Import** option from the **File** menu. Otherwise, we might make mistakes while specifying all the details about the physical tables.

2. Circular joins shouldn't be used in the repository. Create aliases for each physical table to avoid the circular joins. You may create more than one alias for a single physical table.

3. Be sure that physical joins exist in the Physical layer. Although it's possible to import the foreign key relations from the database automatically, we have to be sure that foreign keys exist. If not, we will have to create physical joins in the Physical layer. These relations are going to be used during generation of the physical SQL statements. Open the Physical Diagram and check the relations.

4. Configure the **Features** settings in the **Database** window. If the database doesn't support some of the features, we can easily disable them in the **Features** tab of the database properties window.

5. Set the value of the **Maximum Connections** settings in the **Connection Pool** properties. We have to measure the performance of the database before changing this value. If there's no bottleneck in the database then it will be better to increase the default value. The default value is set to **10**.

6. Create separate databases and connection pools for security, as displayed in the following screenshot:

How it works...

The Physical layer is the first layer that we're going to configure. All the others depend on this layer. Obviously having a well-designed data warehouse model will make the implementation easy. The star schema or the snowflake schema models should be considered in the design phase. Whenever a business user runs an analysis, the Physical layer objects will be used to access the data.

Best practices of the Business Model and Mapping layer

The Business Model and Mapping layer is responsible for the business rules and it depends on the Physical layer objects. The BI server is going to generate SQL statements based on the BMM layer objects. Obviously, the objects defined in this layer may positively improve the performance. We have to design all the BMM layer objects carefully. Here are the some types of the objects that are used in this layer:

- ▶ Logical dimension tables
- ▶ Logical fact tables
- ▶ Logical table sources
- ▶ Logical dimensions
- ▶ Calculated measure columns

How to do it...

1. Although the BMM layer objects are transparent to the end users, one should create logical dimension and fact tables with meaningful names and use prefixes to distinguish them from each other.

2. Ensure that the levels of hierarchies are set to logical columns correctly. All the dimension levels should be mapped with the logical table columns. In the case of unmapped levels, business users are going to have issues when they drill down through that level.

3. Set the aggregation rule for each logical measure column in the fact tables. Otherwise, the output of the analysis is not going to be an aggregated (summary) result set.

4. Ensure that logical joins are created. We can check them by using the Business Model Diagram. Logical joins are going to be created automatically based on the physical joins that are defined in the Physical layer.

5. If any change is needed in the logical join attributes, for instance, if the business requirements need to have outer joins, change the **Type** property from **Inner** to **Outer**.

6. Set the number of elements in the dimension hierarchies. This setting will affect the query performance. It doesn't need to be accurate but at least it should reflect the actual ratio. The BI server decides the usage of the aggregate tables by checking this value. If there are two aggregate tables that can satisfy the logical query, then the BI server is going to use the aggregate table that contains the smaller number of elements.

How it works...

The BMM layer contains objects that are going to be used by the BI server during query generation. Whenever an analysis is executed, the request is going to be sent to the BI server. The BI server is going to convert the logical SQL statement to the physical SQL statement based on the definitions of the BMM layer objects.

We can improve the performance of the queries by creating well-designed business models in this layer. If we don't, or if we make mistakes in this layer, the analysis is not going to produce an accurate result set.

There's more...

Use the global consistency check tool to find out if there are warnings or errors in the repository. Global consistency check is automatically triggered when you try to save the repository. You can also access it from the **File** menu.

Best practices of the Presentation layer

The Presentation layer holds the subject areas, presentation tables, and presentation columns. They are all exposed to the end users through the Presentation Services (web user interface). All the objects in this layer depend on the BMM layer objects.

Subject areas should be created based on **Roles**. We should always think from the user perspective when it comes to designing the subject areas. We can also set the permissions on the objects so that every user logged in to Presentation Services won't be able to access them.

How to do it...

1. Create role-based subject areas based on the business requirements. You can see the sample design in the following screenshot:

2. We should list the dimension tables first, and then the fact tables will be at the bottom of the list. Also it's better to have the time presentation table first in the order.

3. By Default, Presentation Table names are inherited from the BMM Layer. Use meaningful names for the **Presentation Tables** and write a description for each of them in order to assist the end users instead of using the default names.

4. Avoid using many presentation columns in the presentation tables. This will cause end users to lose concentration. Try to use a small number of columns in these tables. It will be good to set up the order of the columns as well.

How it works...

All the subject areas defined in the Presentation layer will be displayed in the user interface. When a business user tries to access the Analysis Editor, the subject area selection page will be prompted to pop up in the screen. Subject areas will be displayed as a list based on the user's privileges. If the user hasn't been granted with the read privilege then that subject area is going to be hidden in the list.

There's more...

We can also configure the security settings at the table and column level. Although the users have sufficient privileges to see the subject areas, they might not see all the presentation tables or the presentation columns. This behavior depends on the **Permission** settings at the proper level.

Best practices of the analyses and the dashboards

The queries that are executed against the data warehouses cause a large amount of database processing. Every time we access an analysis, the query is going to be generated by the BI server and it's executed against the database. Let's assume that a dashboard contains many analyses that will cause intensive calculations. When the business users access that dashboard, all the analyses will run. This will impact the performance. We should not publish many objects in the main dashboards. The users should see the results as quickly as possible. We can achieve this by publishing summary result sets in the main dashboards. If they are interested in the finer granular levels, they can drill down or navigate to another analysis that contains the details.

How to do it...

1. Create the dashboards based on the roles or the functionality area.

2. Use the Dashboard Pages to divide the content, to enable end users to focus on the data.

3. Avoid designing dashboards that will generate large result sets. It's going to have a negative impact on the performance. If the users are interested in more details, they could drill down to more detailed levels through the main reports.

4. Try to use fewer columns in the analyses to not lose focus. Simple designs are mostly the best.

How it works...

Once the users are logged in to the Presentation Services, their home pages are going to be displayed and all the analyses on the home pages are going to run. So it's better to have simply-designed dashboard pages as home pages. When users navigate to any other dashboard page, the same behavior should be the case. We shouldn't publish many analyses on one dashboard page.

There's more...

We can also gain the benefit of the navigation links that are mapped to the analyses. Instead of publishing the analysis, links will be displayed on the pages. Publishing the presentation catalog folders will have a similar positive impact on performance.

Performance and security tips

Business users shouldn't access all of the analyses in Presentation Services. We should set the permissions of the catalog objects so that users can only access the analyses that they need from a business perspective. We can enforce security settings in the repository and also in the Presentation Catalog.

Besides security, query performance is another issue in the BI projects. Whenever analyses run, they should display the results quickly. For instance, generating one result set in an hour is not acceptable. We should tune the performance in such cases to satisfy the business goals.

How to do it...

1. Grant only the privileges required for business users in Presentation Catalog. You can set permissions for all of the catalog objects. Just select the required privilege from the **Permission** drop-down list.

2. Use data filters for the users to generate a limited result set based on the user needs. This can be achieved in the repository. Open **BI Administrator Tool** and access Identity Manager from the **Manage** menu.

3. Use the aggregate tables to improve the query performance.

4. Whenever business users run the analyses, the query will be generated by the BI server and it's going to be executed at the database level. In order to improve the query performance, we should balance the workload by enabling the cache on the BI server. So when the query is executed for the first time, the result set is going to be cached on the BI server. The result set is going to be generated from the cache in subsequent executions. In order to gain the benefit from this feature, caching should be enabled on the BI server. You can modify this setting in the **Enterprise Manager 11g Fusion Middleware Control** window.

How it works...

After configuring the security settings, the users will be able to access the required catalog objects. Depending on the permissions, they will see some of the objects. Another important security point is configuring the data level security. What if two different users have read permissions on the same object and they need to see different data depending on their roles? So once the users are logged in to the web interface, catalog permissions will be applied. Then when they run a report, the security settings defined in the repository will be applied.

The Major Components of OBIEE 11g

In this chapter, we will cover:

- ▸ The major components of OBIEE 11*g*
- ▸ Sample processing of an analysis

Introduction

OBIEE 11*g* consists of several major components and it uses the **Fusion Middleware Server** and the **WebLogic Server**. It's important to understand the functions of these components. We're going to use **Enterprise Manager 11g Fusion Middleware Control** to manage the components.

The major components of OBIEE 11g

After the installation of OBIEE 11*g*, you can access the **Fusion Middleware Control** from the address `http://ServerName:7001/em`. The default port of the Weblogic Server is 7001 and it can be updated to any available port by modifying the `<listen-port>` node of the `config.xml` file.

We're going to discuss these components in this appendix:

- ▸ BI Presentation Services
- ▸ BI Server
- ▸ BI Scheduler
- ▸ BI Cluster Controller

▸ BI Java Host

Major components

▸ **BI Presentation Services**: This is the user interface that provides the BI data to the web clients. Presentation Services is dependent on the BI server. All of the requests are going to be sent to the BI server. You can have access to Presentation Services from `http://ServerName:7001/analytics`.

A repository of Presentation Services is stored on the file system as a folder. A sample path is specified as follows:

```
<ORACLE_INSTANCE>\bifoundation\
OracleBIPresentationServicesComponent\coreapplication_obips1\
catalog\SampleAppLite
```

All of the objects that you're going to create will be stored in the catalog folder.

▶ **BI Server**: This is the core service of the suite. It is responsible for converting the logical request to a physical SQL statement and executing it on the data warehouse. Its metadata is stored in the repository file that is called an RPD file. We can use **BI Administration Tool** to modify the content of this RPD file.

▶ **BI Scheduler**: This component is responsible for running scheduled tasks such as agents or scripts. **Job Manager Tool** can be used to manage the BI Scheduler Service. You can see an example of an agent configuration in the following screenshot.

▶ **BI Cluster Controller**: This is a component that is responsible for distributing the requests to the BI server instances to load the balance. If you've installed clustered BI servers then this service is going to ensure load balancing.

▶ **BI Java Host**: This component is responsible for displaying graphical objects such as gauges and graphs.

Sample processing of an analysis

When business users log in to Presentation Services and execute one of the analyses, some processing steps are executed in the background. We're going to cover these steps in this recipe.

Processing steps

1. Web client runs the analysis to display the result set.

2. Presentation Services builds the logical SQL statement and forwards this statement to the BI server.

3. The BI server converts the statement to a physical SQL statement and executes it on the database.

4. The database server generates the result set and sends it to the BI server.

5. The BI server forwards the result set to Presentation Services.

6. Presentation Services displays the formatted result set so the web client accesses the report result.

ORACLE **Business Intelligence**

	Requested Quantity	Shipped Quantity
Customer		
⊟ Customer Total	2,788,062	2,720,119
⊞ Central	546,663	529,587
⊞ East	1,191,635	1,166,586
⊞ West	1,049,764	1,023,946
Grand Total	**2,788,062**	**2,720,119**

There's more...

If the caching option of the BI server is enabled, then the result set is going to be stored in the cache. In the next execution of the analysis, the query will be satisfied from the cache instead of executing the physical SQL statement on the database.

The caching option is controlled in the **Fusion Middleware Control** window, as displayed in the following screenshot:

Index

Proudly sourced and uploaded by [StormRG]

[PACKT] enterprise
PUBLISHING
professional expertise distilled

Thank you for buying
Oracle Business Intelligence 11*g* R1 Cookbook

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution-based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.PacktPub.com`.

About Packt Enterprise

In 2010, Packt launched two new brands, Packt Enterprise and Packt Open Source, in order to continue its focus on specialization. This book is part of the Packt Enterprise brand, home to books published on enterprise software – software created by major vendors, including (but not limited to) IBM, Microsoft and Oracle, often for use in other corporations. Its titles will offer information relevant to a range of users of this software, including administrators, developers, architects, and end users.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to `author@packtpub.com`. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.
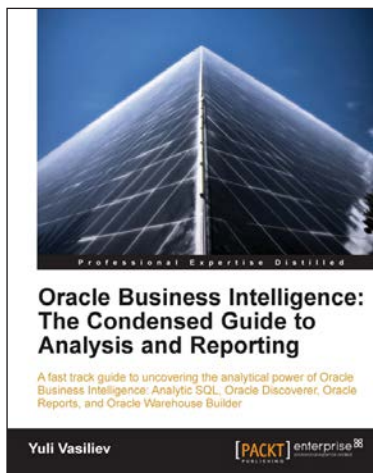
Business Intelligence Cookbook: A Project Lifecycle Approach Using Oracle Technology

ISBN: 978-1-84968-548-1   Paperback: 368 pages

Over 80 quick and advanced recipes that focus on real-world techniques and solutions to manage, design, and build data warehouse and business intelligence projects

1. Full of illustrations, diagrams, and tips with clear step-by-step instructions and real time examples to perform key steps and functions on your project

2. Practical ways to estimate the effort of a data warehouse solution based on a standard work breakdown structure.
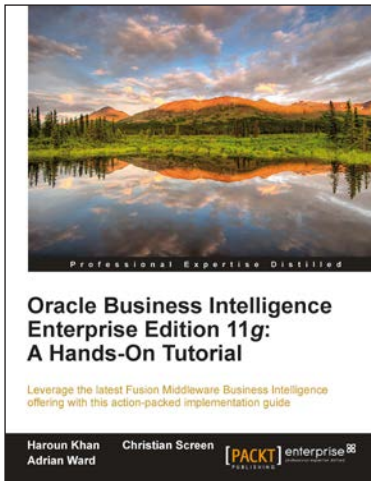
Oracle Business Intelligence : The Condensed Guide to Analysis and Reporting

ISBN: 978-1-84968-118-6   Paperback: 184 pages

A fast track guide to uncovering the analytical power of Oracle Business Intelligence Analytic SQL, Oracle Discoverer, Oracle Reports, and Oracle Warehouse Builder

1. Install, configure, and deploy the components included in Oracle Business Intelligence Suite (SE)

2. Gain a comprehensive overview of components and features of the Oracle Business Intelligence package
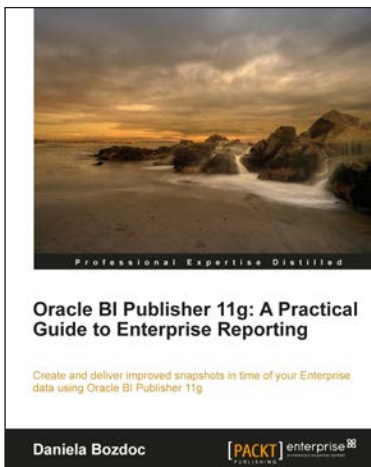
Please check **www.PacktPub.com** for information on our titles

[PACKT] enterprise
professional expertise distilled

Oracle Business Intelligence Enterprise Edition 11*g*: A Hands-On Tutorial

ISBN: 978-1-84968-566-5   Paperback: 620 pages

Leverage the latest Fusion Middleware Business Intelligence offering with this action-packed implementation guide

1. Get to grips with the OBIEE 11g suite for analyzing and reporting on your business data

2. Immerse yourself in BI upgrading techniques, using Agents and the Action Framework and much more in this book and e-book

3. A practical, from the coalface tutorial, bursting with step by step instructions and real world case studies to help you implement the suite's powerful analytic capabilities

Oracle BI Publisher 11g: A Practical Guide to Enterprise Reporting

ISBN: 978-1-84968-318-0   Paperback: 254 pages

Create and deliver improved snapshots in time of your Enterprise data using Oracle BI Publisher 11g

1. A practical tutorial for improving your Enterprise reporting skills with Oracle BI Publisher 11g

2. Master report migration, template design, and E-Business Suite integration

3. A practical guide brimming with tips about all the new features of the 11g release

Please check **www.PacktPub.com** for information on our titles