INFS3200 Advanced Database Systems

# **Prac 2: Data Warehousing (5%)**

### Semester 1, 2021

**Introduction**

**Learning objectives:**

* Learn how to create a cube and its dependent components in Oracle OLAP, including:
  + Identify and build the dimensions
  + Define measures (both stored and calculated)
* Learn how to map the OLAP model to source data and load data into the dimensions
* Learn how to view the OLAP data
  + Query through hierarchies
  + Perform operations like roll-up, drill-down, and pivot
* Understand how data is organized and stored in dimensions and data cubes
* Learn how to query a data cube using SQL queries
* Understand how materialized views are used in cube query optimization

**Assessment:**

*Due at week 9 (online submission)*

This prac carries 5 marks.

Marking Scheme:

* 1 mark: Complete the cube creation in Task 1.
* 1 mark: Maintain the cube successfully in Task 2.
* 2 mark: Complete Task 3, each bullet point is worth 1 mark.
* 1 mark: Complete Task 4.

Screenshot the data cubes you created as well as the results of your OLAP operations, please include your screenshots, SQL queries and necessary descriptions in your submitted document. Please make sure your screenshots contain your student ID as the proof of originality. Put all your content in a word/pdf document or leave scripts in separate files and pack all files into a zip/rar package. The file name should contain your name and student ID. Please format your document nicely to help tutor’s marking process. A poorly formatted document may receive a reduced mark. *Submit your work to the Blackboard site by 16:00 pm, April 30th.* Late submission will receive 2 marks’ penalty every 12 hours.

**Preparation: Data Warehouse Setup**

1. **System overview**

**Oracle OLAP**: Oracle OLAP is a multidimensional analytic engine embedded in Oracle Database. It supports online analytics based on data warehousing techniques. In Oracle OLAP, a cube provides a convenient way of collecting stored and calculated measures with similar characteristics, including dimensionality, aggregation rules and so on. A particular analytic workspace (AW) may contain more than one cube, and each cube may describe a different dimensional shape. Multiple cubes in the same AW may share one or more dimensions (Fact constellation, Lecture5-P31).

**Analytic Workspace Manager (AWM)**: In Oracle OLAP, AWM is an easy-to-use GUI tool for creating, developing, and managing multidimensional data in an Oracle data warehouse.

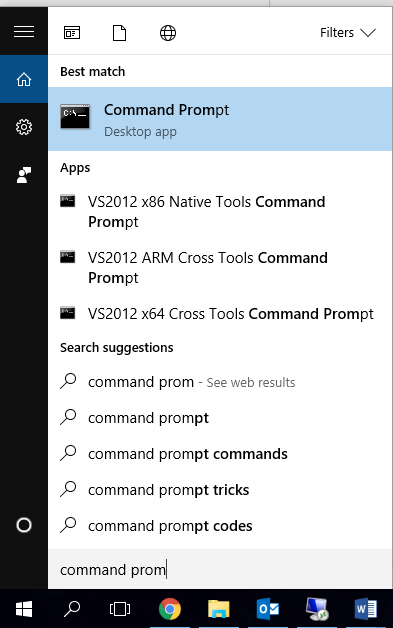
1. **Data description**

OLAPTRAIN is a **star schema** that was sourced from a transactional system, which contains data for a fictional electronics store. The followings are the tables (including four dimension tables and one fact table), intended for analysis, that previously went through the ETL (Extraction, Transformation, and Loading of heterogeneous data) process:

|  |  |
| --- | --- |
| **Table** | **Description** |
| CHANNELS | Table containing distribution channels for customers’ purchases. |
| CUSTOMERS | Table that shows who purchased products, and where products were sold for the Geography dimension. |
| PRODUCTS | Table containing products sold by the company. |
| TIMES | Table containing time periods when products were sold. |
| SALES\_FACT | Stores purchases in dollars, quantity and price, by the channel of distribution, product item, day, and customer. |

1. **Data import**

Download the “P2.rar” package from Assessment on Blackboard and extract it to a local directory, for example “c:\app\P2\”. The package contains the AWM software, the OLAPTRAIN schema, cube templates and query scripts used in the following tasks. Before analyzing the data, we first import the OLAPTRAIN schema into the Oracle database. Open a **Command Prompt** by searching ‘command prompt’ in the search window (Please run it as administrator to avoid privilege issues), shown as below, and complete the following steps:



/\*Enter the directory that has the installation files, in my case it is “cd c:\app\P2\olaptrain\_install”\*/

**>** cd **YOUR\_P2\_FOLDER**\olaptrain\_install

/\*Login to SQL\*Plus\*/

**>** sqlplus sys as sysdba

Enterpassword: Password1!

/\*Set a system parameter to avoid a future error when creating users\*/

**SQL>** alter session set "\_ORACLE\_SCRIPT"=true;

/\*Run the OLAPTRAIN installation script\*/

**SQL>** @install\_olaptrain\_student

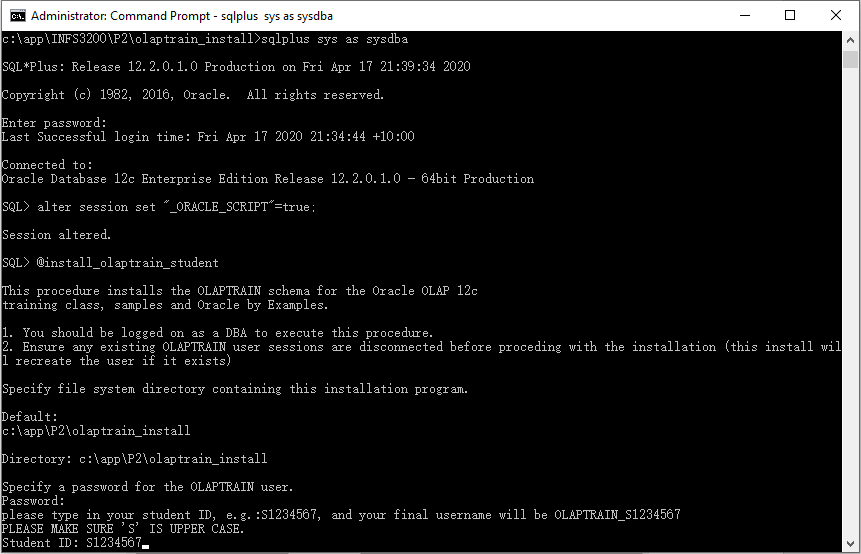
/\*Enter the install directory and password\*/

Directory: YOUR\_P2\_FOLDER\olaptrain\_install

Password: w

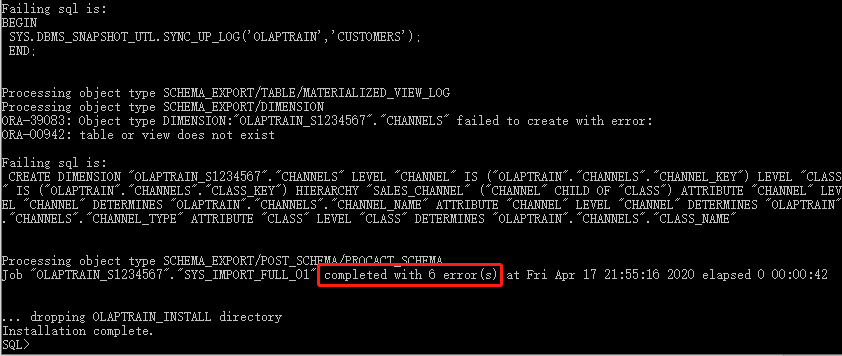
StudentID: S1234567

Note that the directory should be the same as above and you can choose your own password for user “OLAPTRAIN\_S1234567” (Please **DO NOT** include ‘-‘ or other special characters in your password), which will be the main user throughout this practical. Please change “S1234567” to your student ID and make sure “S” is upper case, this is crucial. The processes are shown below:

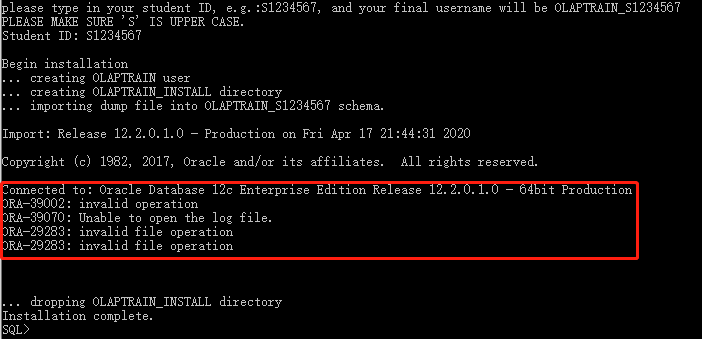


**Troubleshooting:**

A **successful** installation should end up with 6 errors, shown as follows:

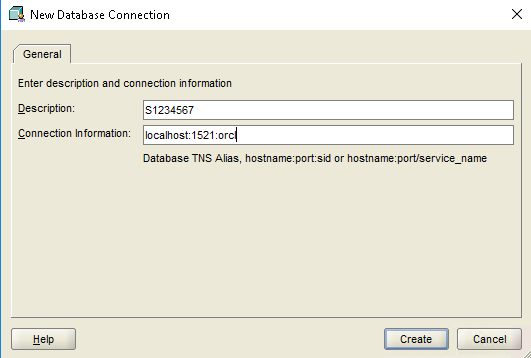


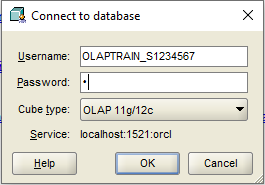
However, if the installation ends quickly with the following messages, it is usually caused by specifying an incorrect directory.

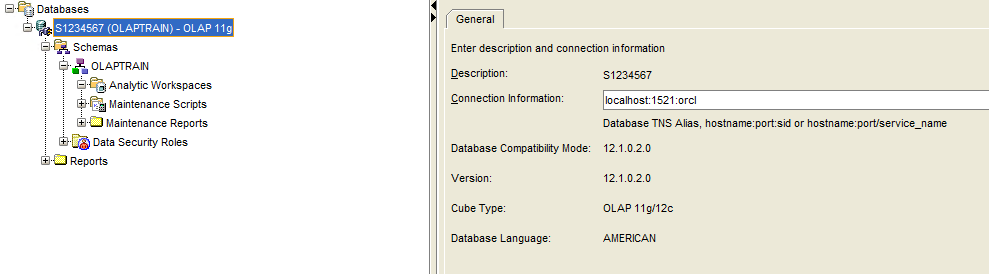


1. **Connect to data warehouse using AWM**

Click the “awm.bat” in “awm122010\_Standalone” folder to open the AWM (if there is no response after a click, please check if you have Java JDK installed in your system, Java 8 recommended). Create a new connection to the Oracle database and set the **Description** as your student ID and the **Connection Information** as ‘localhost:1521:orcl’. Click the connection and log in using the username “OLAPTRAIN\_S1234567” and the password you set.



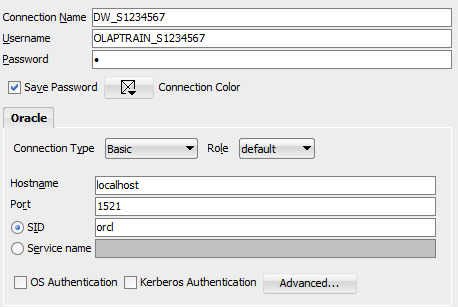




**Part 1: Create Logical Data Model**

1. **Understand the data**

Before designing the data model, it is highly suggested to first understand the *OLAPTRAIN* schema. Use **SQL developer** to connect to the database and check the following tables: *channels, customers, products, times* and *sales\_fact.* The connection name should include your student ID, like “DW\_S1234567”.



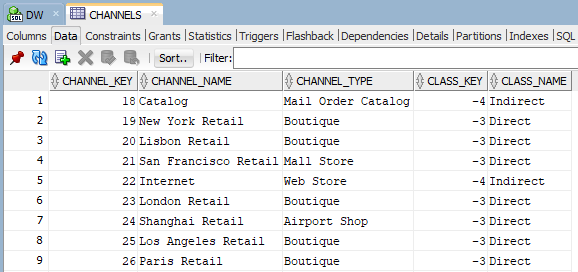
After examining the OLAPTRAIN schema, we need to identify the dimensions, hierarchies and attributes of our data model. In addition, we define various measures based on business interests.

1. **Identifying Dimensions**

Using the source data tables as the primary input, the following dimensions are identified as requirements for the OLAP data model:

* Channel
* Geography
* Product
* Time

Besides, each of the dimensions contains hierarchical structure, for example, in CHANNELS table, shown as follows, we can identify four hierarchies: the actual channels (channel\_name/channel\_key) -> channel type (channel\_type) -> channel class (class\_name/class\_key) -> (not chosen in group-by query, refer to Tutorial5-Q2, here we name it as all\_channel).



1. **Identifying Measures**

The measures are defined based on common business interests, each of which is equivalent to an SQL aggregation query. The measures include both stored and calculated measures. Stored measures are facts acquired from the fact table directly, while the calculated measures require complicated calculations over one or multiple facts. In this dataset, we focus on the following measures:

**Stored Measures**

* Sales
* Quantity

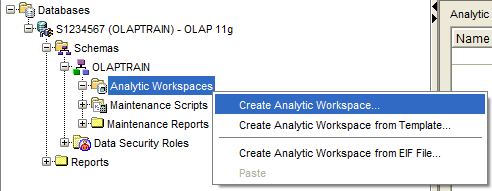
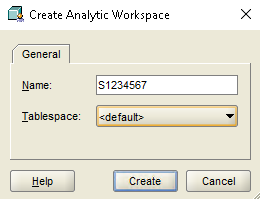
**Calculated Measures**

* Sales Year-to-Date
* Sales Year-to-Date Prior Year

The measures will be defined during the creation of cubes, which will be introduced later.

1. **Create analytic workspace**

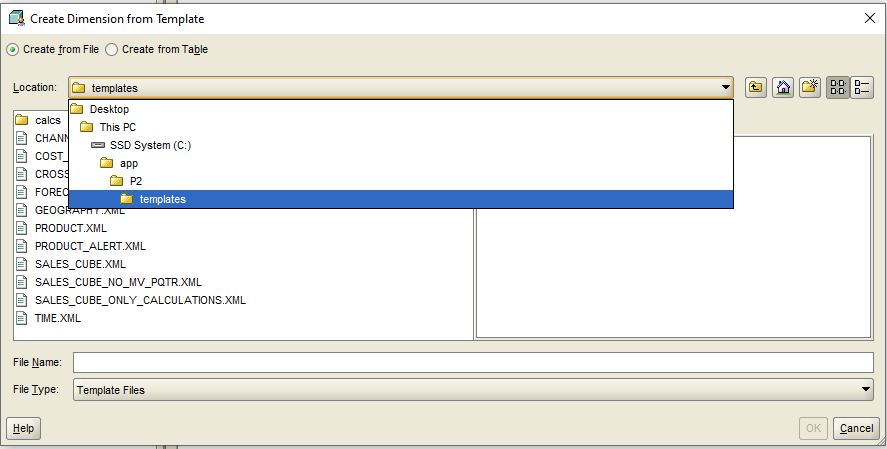
Right-click **Analytic Workspaces** and select **Create Analytic Workspace** to create a new analytic workspace under the name of your student ID.

1. **Create dimensions using templates**

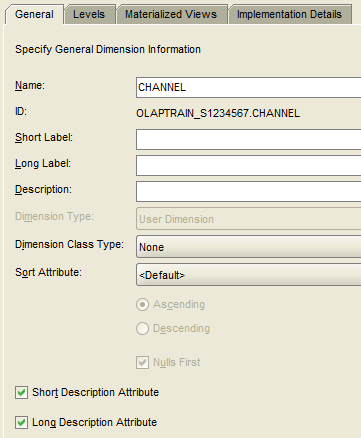
We provide templates for all four dimensions so that you do not need to define them manually. The template feature in Analytic Workspace Manager saves the definition of the OLAP data objects as an XML file. Using a saved template, you can create a new analytic workspace, dimension, cube, and measure exactly like an existing object, with or without mappings. Templates do not include the data, only the definition of the object.

In order to import a template, right-click the **Dimensions** folder, then select **Create Dimension from Template.** The templates are stored in the *templates* folder in the extracted folder.



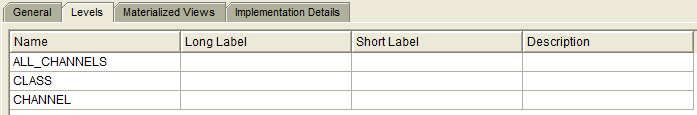
Import *Channel*, *Geography, Product and Time* dimensions in the same way. Check the settings of these dimensions and make sure all members are mapped to the data source (no modification needed). For example, in *Channel* dimension, the template contains the following settings:

Click the **Dimensions** folder and select *Channel*. In the General tab, we can see the dimension name (Channel) and the dimension type (User Dimension).



In the Levels tab, we can see three levels are defined, which correspond to three of the four possible dimensions mentioned above (channel type not included):

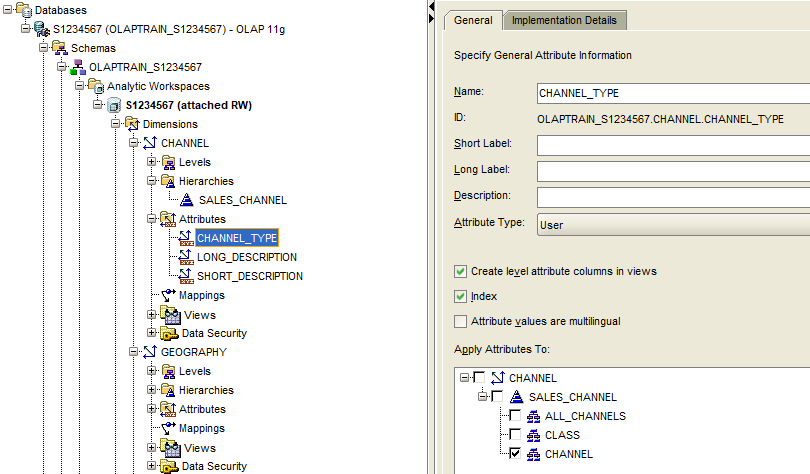
* **ALL\_CHANNELS =**
* **CLASS =** channel class
* **CHANNEL =** the actual channels



Under the Hierarchies in *Channel* dimension, we can see the **SALES\_CHANNEL** hierarchy. These three levels are ordered in the following way, which is consistent with our understanding of the data:

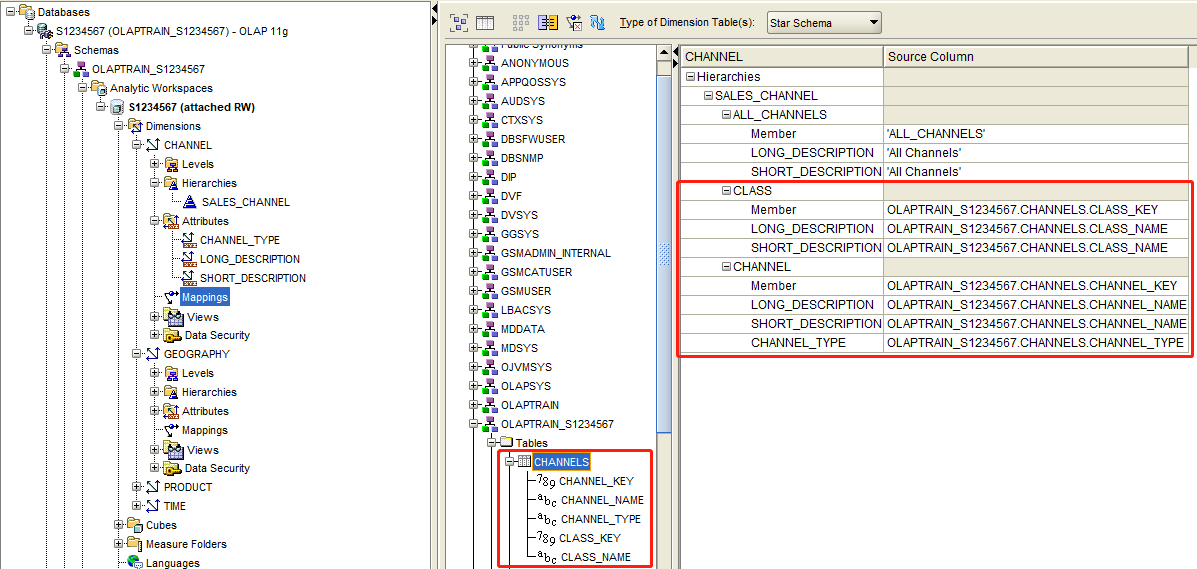


Under the Attribute dialogue, we can see that channel type has been defined as an attribute attached to **CHANNEL** level, instead of being another hierarchy level, which means it is out of the business interest and cannot be queried on.



After defining a dimension, those defined components should also be mapped to the existing tables and views in Oracle Database. Specifically, the *Member* attributes in the OLAP dimension should be mapped to the *key* columns in the dimension tables, while the attribute columns should also be mapped to the appropriate OLAP dimension attributes.

Click the **Mappings** in *Channel*dimension and make sure the *source columns* (left) appears in the *mapping pane* (right) correctly. Note that there is no such source data column for **ALL\_CHANNELS** level, therefore, for "All/Total" hierarchy levels, the descriptions are typed manually:

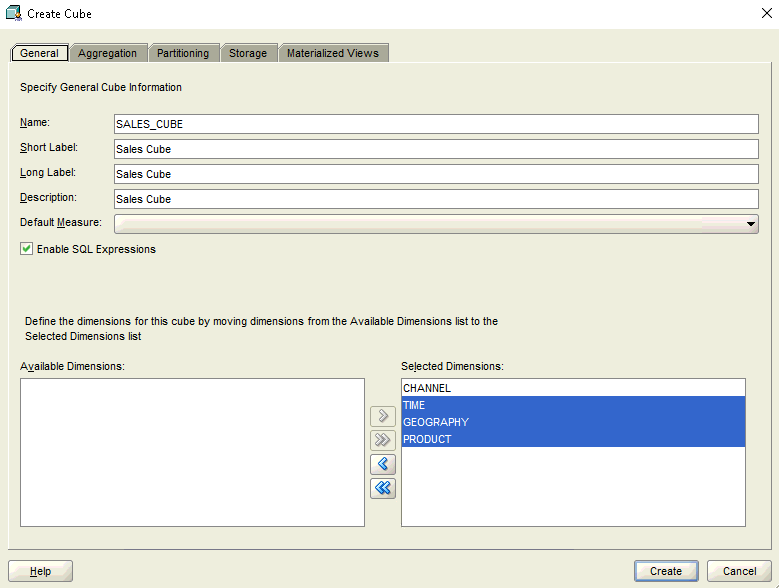


Perform the same inspection to the other imported dimensions and understand how they are defined.

1. **Create Cube**

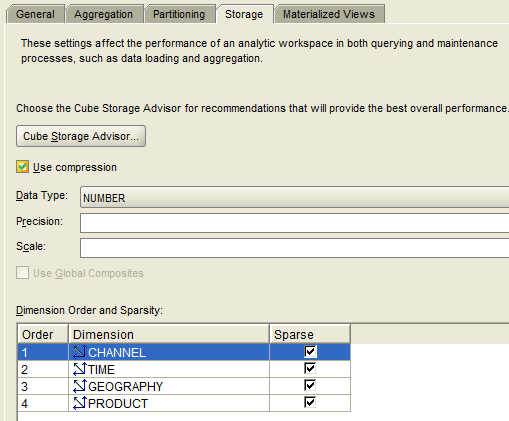
After all dimensions are defined, we are ready to create our cube. Right-click the **Cubes** folder, then click **Create Cube**. In the General tab of the **Create Cube** window, specify the following:

* Name: SALES\_CUBE
* Use the Add button (>) to select dimensions in the following order:
* CHANNEL
* TIME
* GEOGRAPHY
* PRODUCT



**Notes:** The order in which the dimensions are listed in a cube may affect performance because it determines the way the data is stored on disk.

Next, select the Storage tab. The Storage tab helps you manage the data compression strategy. By default, we choose **Use compression**, and then enable the Sparse option for all dimensions, as shown below:



Finally, click **Create** to finish the dialogue.

1. **Create measures**

You can create two types of measures in a cube: *Stored* (or Base) measures, and *Calculated* measures. Every measure that belongs to a particular cube shares the characteristics that were defined for the cube.

***Stored Measures***

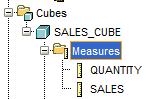
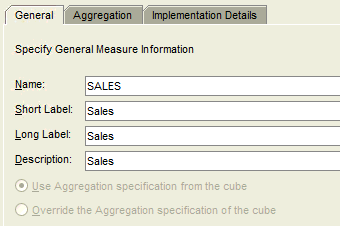
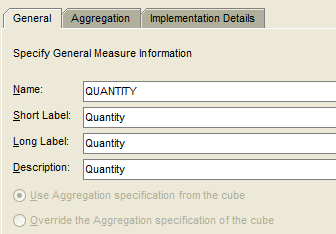
Stored measures store the facts collected about your business. When you create stored measures in your OLAP data model, you will map them to the source data just like what you have done with dimensions.

***Calculated Measures***

One of the powerful features of the Oracle OLAP technology is the ability to efficiently and easily generate business calculations of data held in the database. In any OLAP implementation, the number of calculated measures greatly exceeds the number of stored measures.

OLAP calculated measures are derived from stored measures or other calculated measures. These calculations are computed dynamically when users query the data. Calculations are automatically exposed as columns in a cube view, just like the facts.

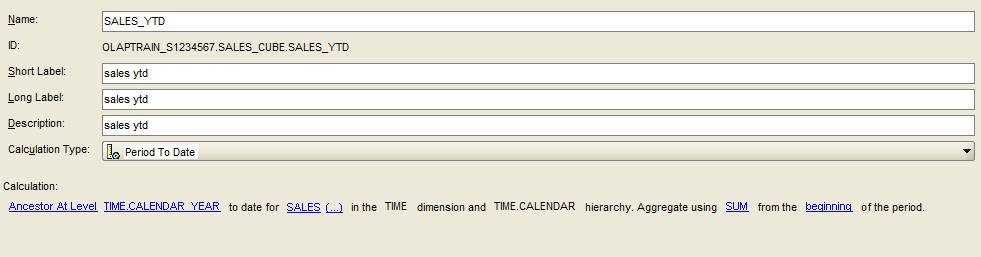
According to our design, we define two measures, i.e., *Sales* and *Quantity*, and two calculated measures. Two measures are created as follows:

And we create the first calculated measures by entering or selecting the following:

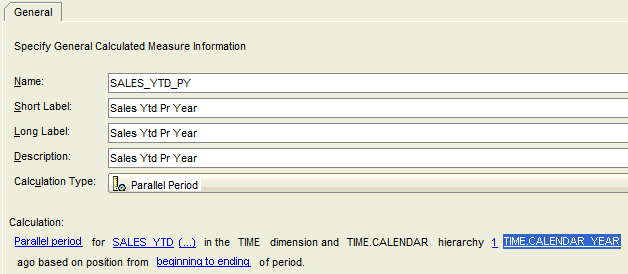
* Name = **SALES\_YTD** (the Label and Description fields are auto-filled)
* Calculation Type = **Period to Date**
* In the Calculation inputs section, select the following:
* First hyperlink = **Ancestor At Level**
* Second hyperlink = **TIME.CALENDAR\_YEAR**

Finally, the window should look like this:



Similarly, we create a YTD calculation for the previous year. This measure facilitates year-to-year comparisons. The settings are shown below:

* Name = **SALES\_YTD\_PY**
* Calculation Type = **Parallel Period**
* In the Calculation inputs section, select the following:
* Second hyperlink = **SALES\_YTD**
* Fifth hyperlink = TIME.CALENDAR\_YEAR

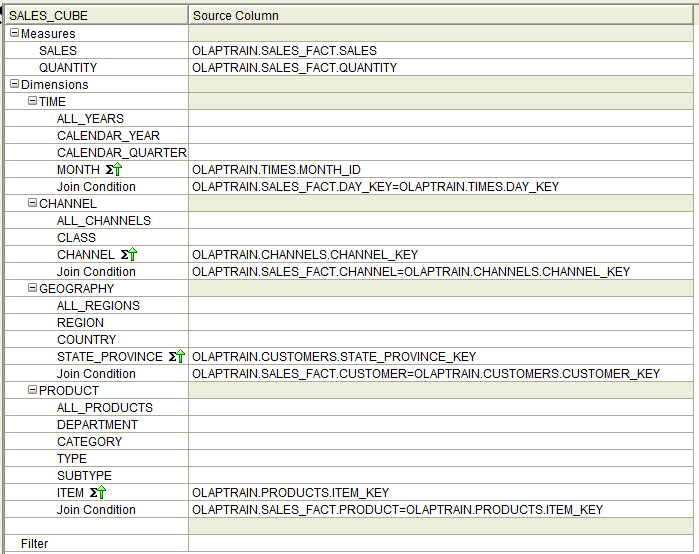


1. **Map the cube**

Same as the dimension mapping, we need to map our cube to the existing data source. In a data cube, we need to map the following fields:

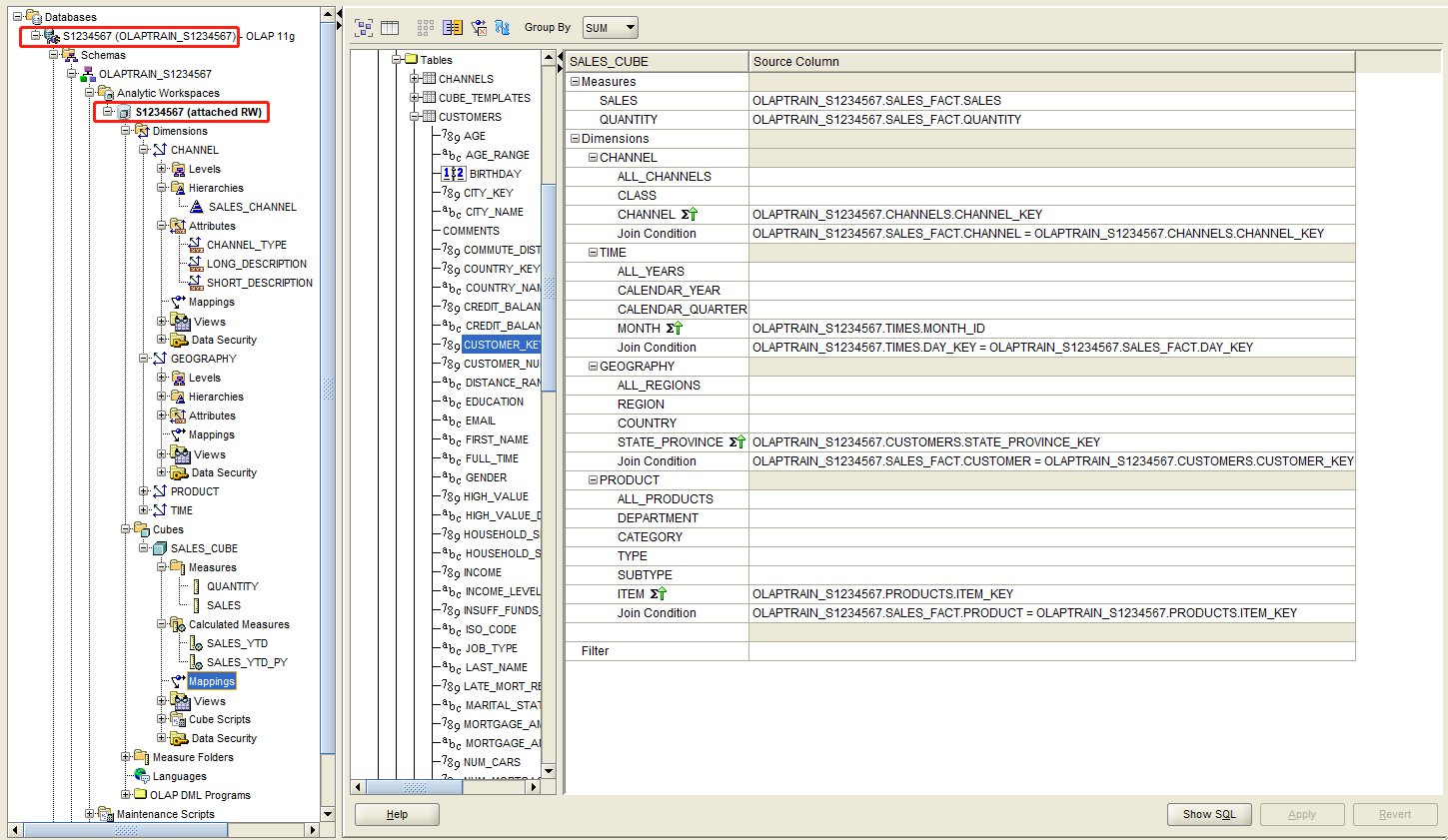
* The stored measures that are defined within the cube.
* The lowest level of detail for each dimension hierarchy.
* The Join Condition field. This field associates the foreign key column from the fact table to the primary key column from the dimension table.

Note that the mapping is done by dragging the corresponding column from the *source columns* (left) to the correct spot in the *mapping pane* (right). A Join condition can be achieved by dragging both joining columns to that slot and the “=” will be added automatically. Please DO NOT type those values manually as it will cause unexpected problems in the future. Eventually, the mapping result should look like the follows. Click **Apply** to complete the mapping.



***Task 1***:Complete all steps introduced above until you successfully create and map the cube. *Take three screenshots of the cube you create, including the mapping, the definition of two calculated measures. Include those screenshots in your document submission.*

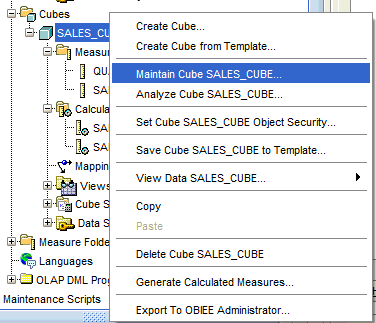
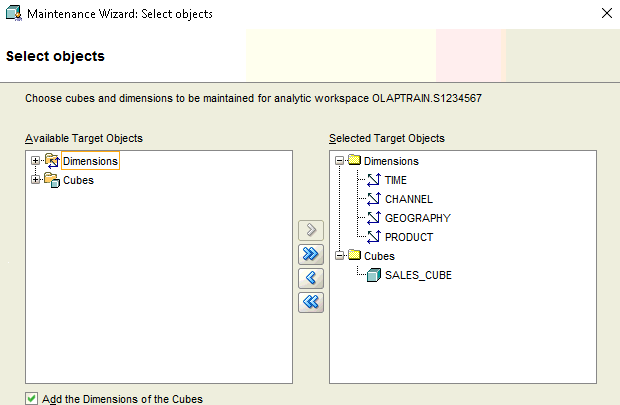
**NOTE:** Please make sure **your student ID** appears in every screenshot you take to show the originality of your work. This rule applies to **ALL screenshots** taken throughout this prac unless specified. Screenshots without student ID may be regarded as invalid and receive reduced marks. An example screenshot of the mapping should look like the follows:



**Part 2: Load and View Cube Data**

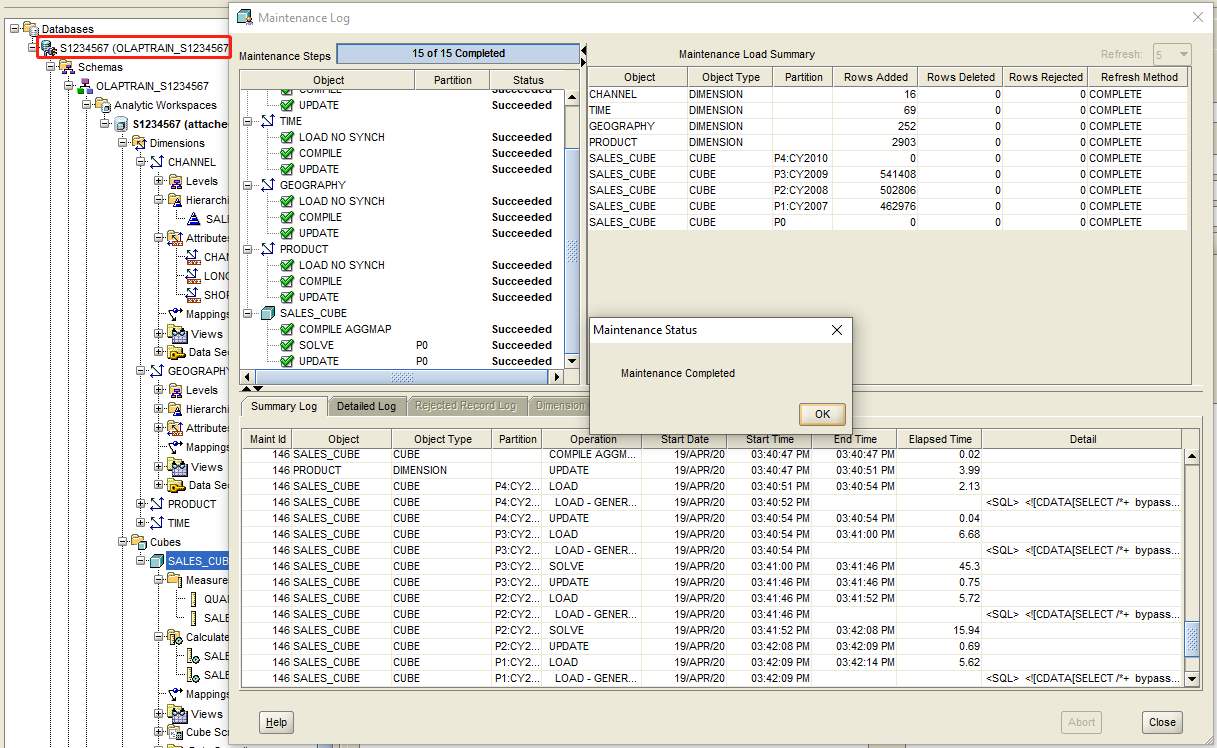
The Maintenance Wizard loads and aggregates the data in a single step. We can load all mapped objects in the analytic workspace, or individual dimensions and measures. We can also choose to run the job immediately, enter it in the Oracle Job Queue, or save it as a SQL script. The materialized views, calculated measures are preprocessed during this maintenance phase. After the maintenance, we can use SQL queries to retrieve the result. Also, we can view the cube data using operators like Roll-up, Drill-down and Pivot.

1. **Maintain the cube**

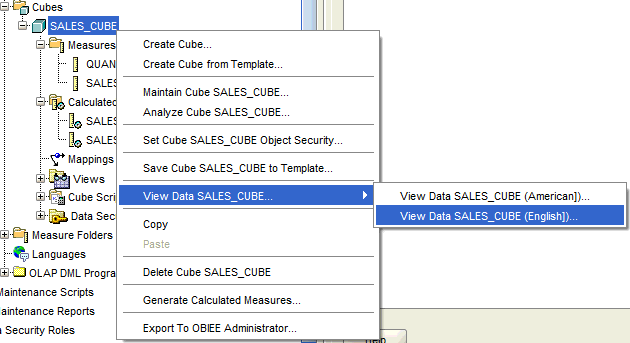
The cube maintenance will load data to the cube. By default, the dimensions of that cube are also processed before the cube. If you have already loaded dimension data, you can specify only to load measure data.

***Task 2***: We start to maintain the cube. It will take quite a while to finish the maintenance. *Please take a screenshot of your maintenance result, same as below, and include it in your document.*



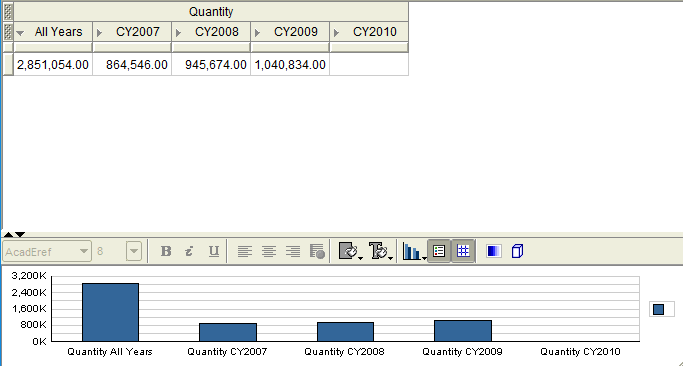
1. **View the cube data**

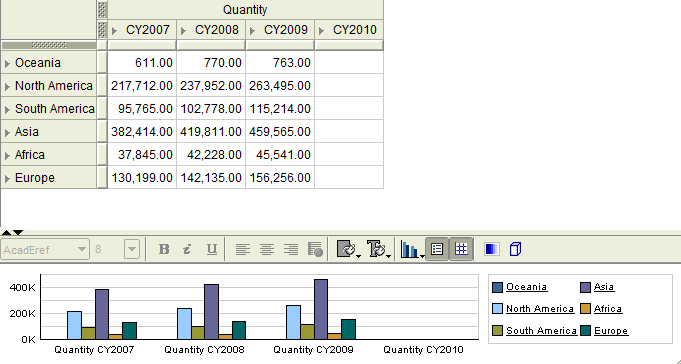
Click the **View Data SALES\_CUBE** to start the Data Viewer.

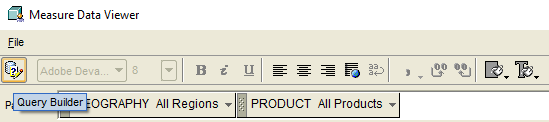


***Task 3***: Now you can play with this tool to explore the data. Meanwhile, you are required to achieve a few goals:

* Perform roll-up, drill-down and pivot operations in Data Viewer, respectively (Lecture5-P36, Tutorial4-Q2). Each operation includes three parts: (1) the screenshot of the view before the operation, (2) a sentence describing which operation you are performing on which columns/levels; (3) the screenshot of the view after the operation. Therefore, there are 6 screenshots and 3 lines of description in total. *Include the screenshots and descriptions in your document.*
* Adjust your Data Viewer window until it is identical (the value can be different) to the following two views (**Hint:** your viewer windows must be identical to get full mark and you may need help from the **Query Builder**, shown below). *Take a screenshot for each view you made and put them in your document.*





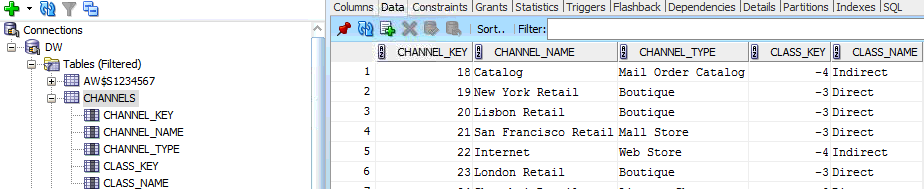


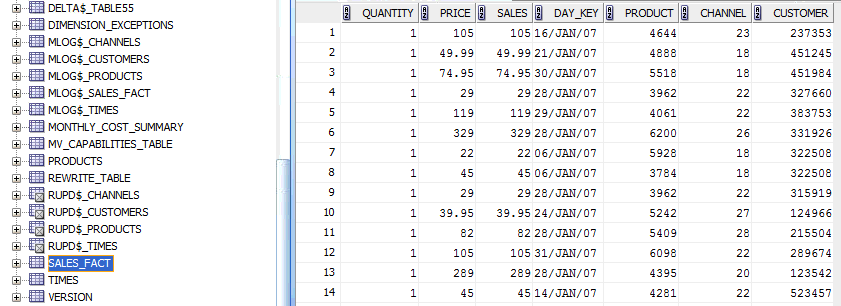
**Part 3: Understand Data Warehouse Design Mechanism**

We have learnt how to create dimensions and cubes using Oracle OLAP and AWM. Now, we will try to understand the mechanism behind. Specifically, there is **NO** **mark** in this part, but it will help you understand the whole process of how data cubes are built, queried and managed by Oracle OLAP, which will make the subsequent tasks easier to follow. Therefore, this part is optional but recommended. Please follow the instructions below.

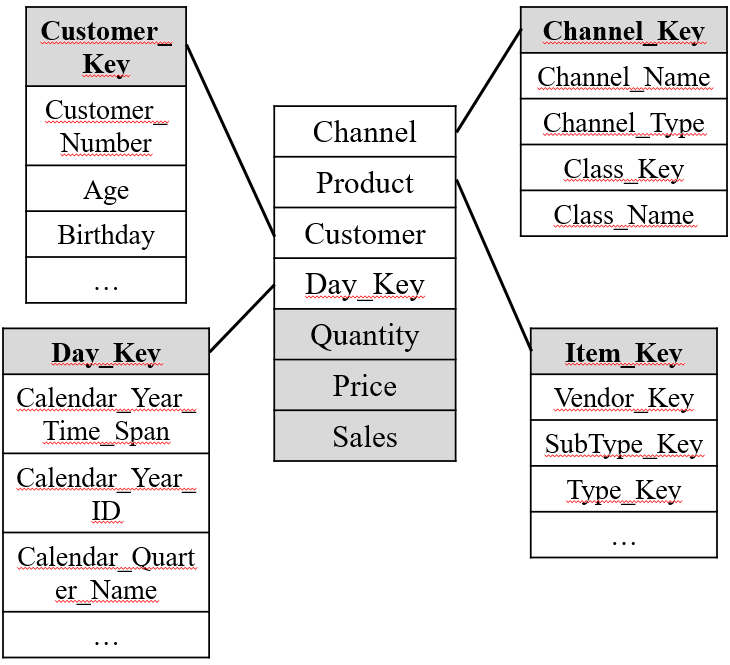
1. **View the dimension tables and fact table**

Connect to the olaptrain schema via **SQL Developer** using your “OLAPTRAIN\_S1234567” (Once again, S1234567 refers to your student number) user, and then check the tables “CHANNELS” and “SALES\_FACT”.





Just like what we have learnt from the lecture and tutorial, the dimension table “CHANNELS” contains dimension key (Channel\_Key), hierarchy (Channel –> Class -> All Channels) and attributes (Channel\_Name, Channel\_Type), while the fact table consists of the dimension key for each dimension (Channel, Product, Customer, etc.) and the facts (Quantity, Price and Sales). The dimension tables and fact table constitute a star schema as follows:

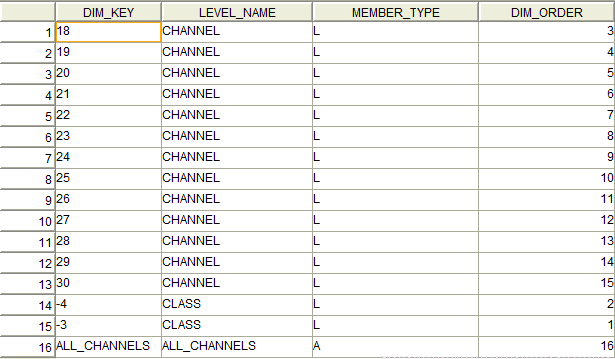


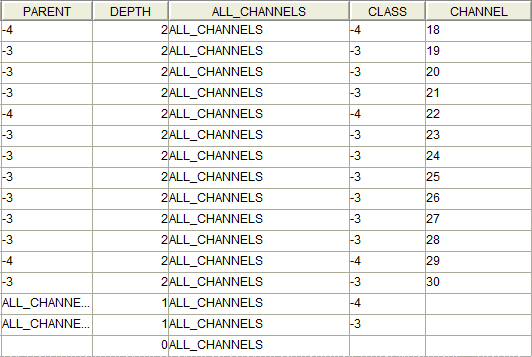
1. **Compare the views with dimension tables and fact table**

Now, let us view the data in the “CHANNEL” dimension in our analytic workspace. Connect to our analytic workspace through **AWM** and open the *views* in the “CHANNEL” dimension. Here we have two views: the dimension view “CHANNEL\_VIEW” and the hierarchy view “CHANNEL\_SALES\_CHANNEL\_VIEW”.

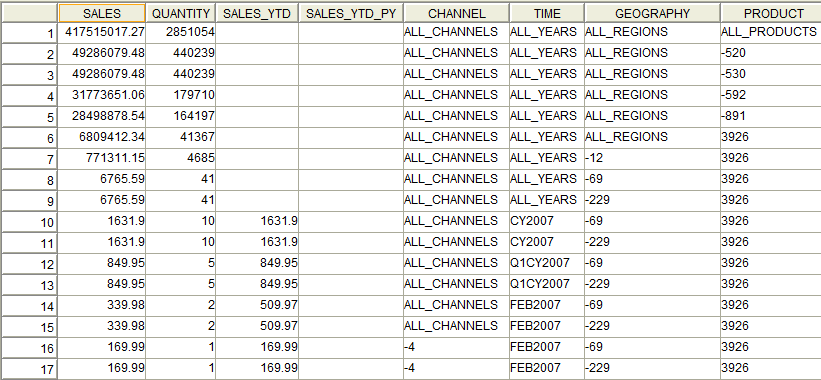


In addition to the raw dimension table “CHANNELS”, the “CHANNEL\_VIEW” stores the hierarchy information, and the “CHANNEL\_SALES\_CHANNEL\_VIEW” contains additional columns indicating the hierarchy level of each record.





By storing the hierarchy information as separate records in dimension tables, we can easily store the summarization of each hierarchy level when maintaining a cube. Check the materialized view “SALES\_CUBE\_VIEW” in the cube “SALES\_CUBE” and see such results:



Therefore, given above materialized views, we can answer the aggregation queries quickly. Assuming the user is viewing the total sales on “All\_Channels”, “All\_Years”, “All\_Regions” and “All\_Products”, we can easily perform a drill-down operation from “All\_Channels” to “Class” by SELECT the Sales from “SALES\_CUBE\_VIEW” where:

* Time = “All\_Years”
* Geography = “All\_Regions”
* Product = “All\_Products”
* Channel in {-3,-4}

as we can identify that -3 and -4 in “CHANNEL\_VIEW” refer to the two possible classes. With the help of the materialized view “SALES\_CUBE\_VIEW” and the dimension tables, the complex aggregation queries are converted to simple selections.

1. **Cube optimization**

With the help of the “SALES\_CUBE\_VIEW”, we can easily get the results of the aggregation queries. However, maintaining such a cube view is not easy. If you open an SQL Plus terminal and type in the following two queries, you will find out that the number of records in the fact table and the cube are 2,811,097 and 24,440,506 respectively (since the second query may take more than one hour, it is not mandatory to run these queries).

/\* Check the count of records in the fact table. \*/

SELECT count(\*) FROM SALES\_FACT;

/\* Check the count of records in the cube view\*/

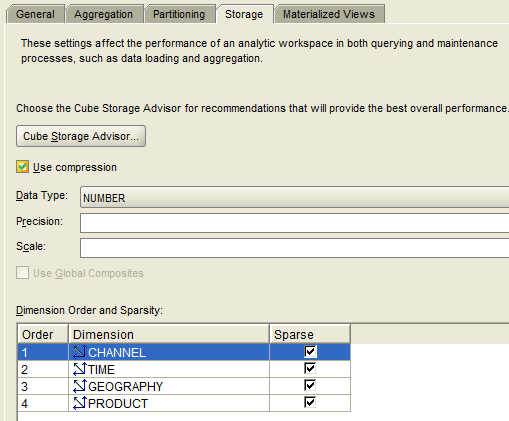
SELECT count(\*) FROM SALES \_CUBE\_VIEW;

Therefore, a cube view contains much more records than the original table since the cube size is determined by the size of dimensions. In our case, although each dimension only contains tens to thousands of rows, the cube size already exceeds the million level. Hence, it is impractical to pre-compute all 24 million of aggregation results in the maintenance step.

In order to achieve a balance between maintenance cost and query performance, the analytic workspace provides a bunch of configurations to control the cost of maintenance and query:

1. **Data compression**

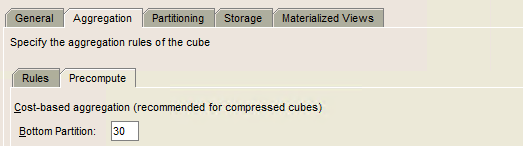
When we built our cube in Part 1, we leveraged the data compression feature and specified the sparse dimensions as shown below:



The data sparsity is very common in dimensional data models. When there are a large number of empty cells in a cube, the cube is said to be “sparse”. In our data cube, if we define multiple dimensions as “sparse” dimensions, the Oracle OLAP will create a special index for the cube so as to automatically manage sparsity. Compression on sparse dimensions can also significantly reduce the size of cubes and improve the performance of both data loads and queries. Do not try to uncheck the “Use compression” and maintain the cube, it will take for days.

1. **Cost-based aggregation**

Cost-based aggregation enables you to balance the maintenance cost and query performance directly. Specify a percentage value and the database will pre-compute and store the most costly aggregate values based on your input. That is to say, if we set the value to 30%, then 30% of the aggregate values will be calculated and stored during data maintenance, and 70% will be calculated in response to a query. Try to change the value into something between 1% to 100%, and you will see a time difference in maintaining the cube. Please maintain the cube again to apply the changes you made each time. However, the query performance varies slightly as the cube is relatively small. (**Note:** “Partition Cube” should be unchecked to enable cost-based aggregation)

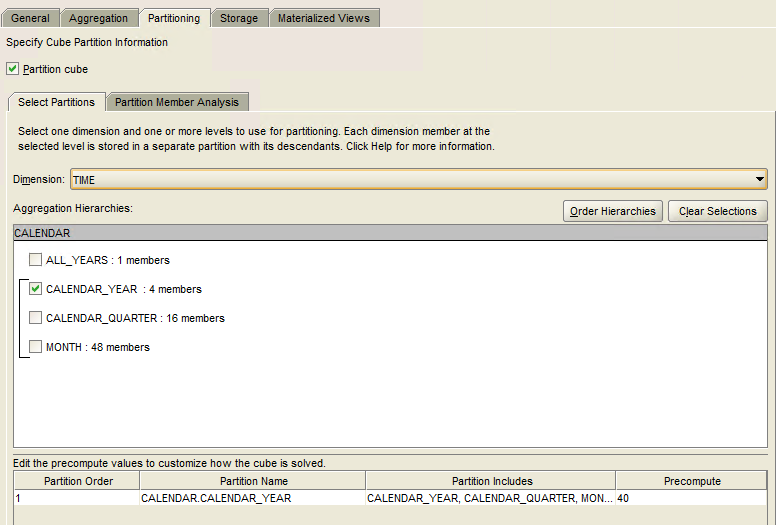


1. **Dimension partitioning and ordering**

Partitioning is a method of physically storing the measures in a cube. It improves the performance of large measures in the following ways:

* Improves scalability by keeping data structures small. Each partition function is like a smaller measure.
* Keeps the working set of data smaller both for queries and maintenance, since the relevant data are stored together.
* Enables parallel aggregation during data maintenance. Each partition can be aggregated by a separate process.
* Simplifies removal of old data from storage. Old partitions can be dropped, and new partitions can be added.

You can activate the partitioning as below, and choose a proper partitioning strategy to accelerate your maintenance and queries.



Also, the order of the dimensions in a cube may affect performance. In general, when you dimension a cube, the first dimension in a cube should have the smallest cardinality and the last dimension has the largest.

**Part 4: Query Data Cube via SQL**

As mentioned above, the views provided by Oracle data cube is similar to traditional table-based star models. However, there are two key differences:

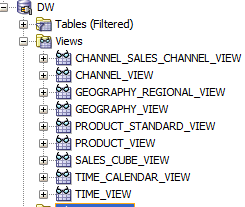
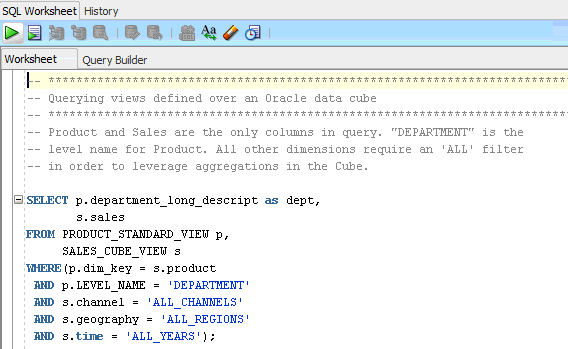
* Fact tables in a star schema store detailed data, while the cube views reveal many summary levels.
* In addition to the facts in fact tables, the cube provides additional measures and calculations, which are calculated and materialized as columns in the cube view.

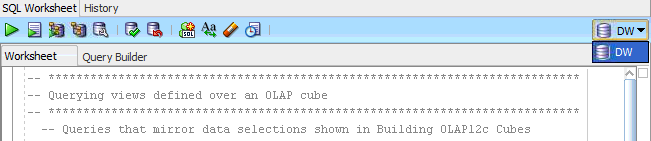
These differences impact the way you query data. With star queries, you aggregate the data by combining aggregation functions (such as sum) and the GROUP BY clause. With cube queries, if the cube has been fully calculated (cube view is fully materialized), you simply select the data you want (either stored or calculated measures) as a column. Typically, no aggregation function is necessary since the data has already been summarized by the cube.

Since the cube data is made directly accessible to SQL by a set of relational views, in this part, we will write some SQL queries to query the cube views.

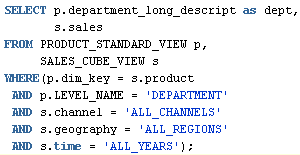
1. **Simple Cube Queries**

Get back to the **SQL Developer** and connect to the olaptrain schema. Check the views that were created in AWM.



Select **File > Open** and open the “cube\_queries.sql” file in the *queries* folder from the extracted files. Check the following query:



Query Notes:

* “Sales” is simply selected as a column. There is no SQL aggregation operation applied.
* A level within the Product dimension hierarchy -- DEPARTMENT -- is used to filter product members.
* All of the dimensions are qualified in the WHERE clause, even though only the Product dimension is selected. In OLAP cube queries, dimensions that are not selected in the query require an “ALL” condition -- which specifies the top-level hierarchy value for each of the dimension columns -- in order to leverage summaries that are already computed by the cube.

Execute the query using your connection (“DW\_S1234567” in the above example). The query should return three rows almost instantaneously. The results should look like this:

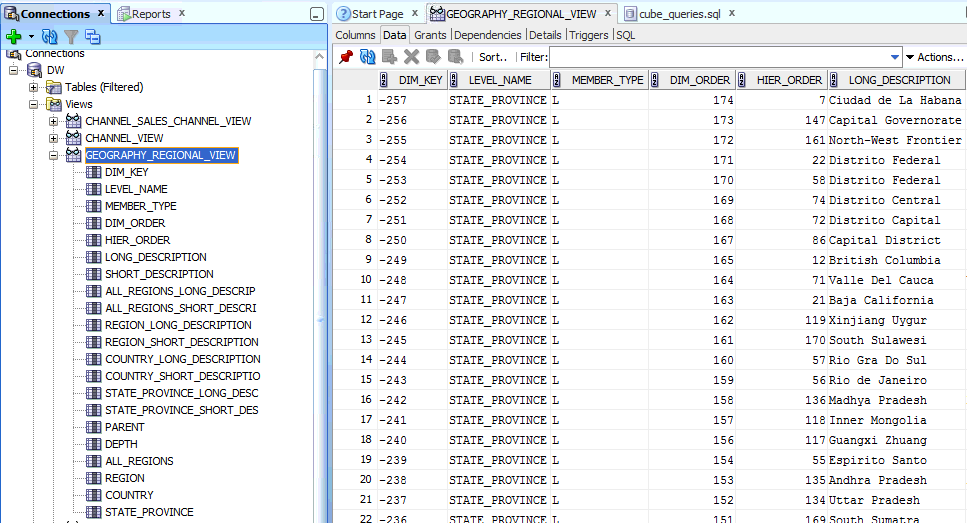


1. **Advanced aggregation query**

Now you are asked to write a similar query as above. In the previous query, a "Level" Condition was used for the Product dimension (which was the only dimension selected). Here, you need to apply level conditions to multiple dimensions in a cube query.

Every hierarchy and dimension view contains a “LEVEL\_NAME” column. The value in this column is the name of the OLAP hierarchy Level object that you created when modelling the dimension in AWM. By simply specifying a value for this column in the WHERE clause, you filter the data to include only those dimension members at the specified level in the hierarchy.

Since the Oracle OLAP truncates column names at 24 characters, in order to get the exact name of a column, simply drill on the view that you want to examine by using SQL Developer’s Connections navigator. You can also view the data to see the exact values in a column.



***Task 4***: Now, complete the following query:

**SELECT** c.class\_short\_description as class,

p.department\_long\_descript as dept,

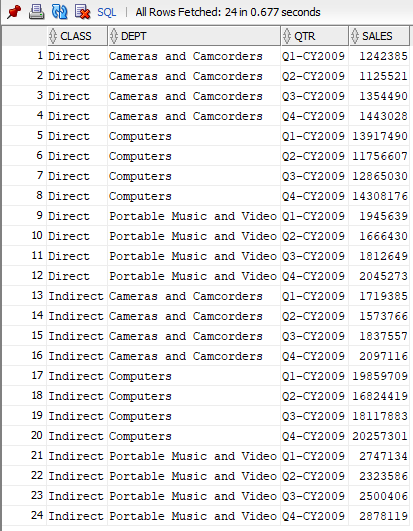
t.calendar\_quarter\_long\_de as qtr,

round(s.sales) as sales

**FROM** ?

**WHERE** ?

Here, *c, p, g, t, s* are the views from *channel, product, geography, time* dimensionsand *sales\_cube,* respectively. Please perform a summarization on dimension levels *channel.CLASS, product.DEPARTMENT, geography.ALL\_REGION and time.CALENDAR\_QUARTER,* and the *time* is filtered for “CY2009” only. The query result should look like this (the record order doesn’t matter):



*Include your query and a screenshot of the results in your document.*