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.

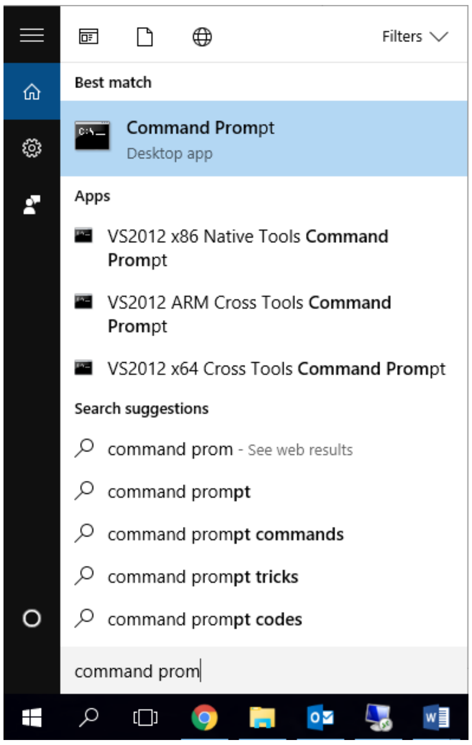
2. Data description

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.

3. 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

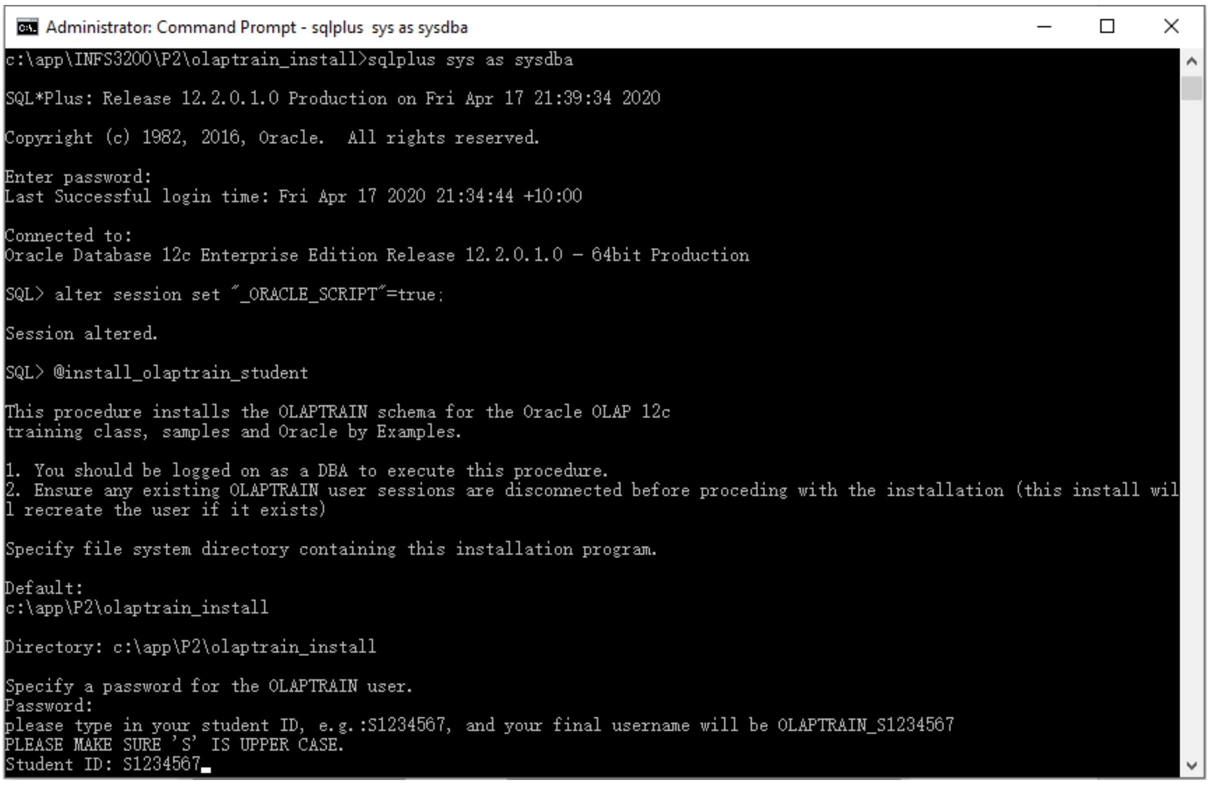
/*Login to SQL*Plus*/
> sqlplus sys as sysdba
Enter password: Password1!

/*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

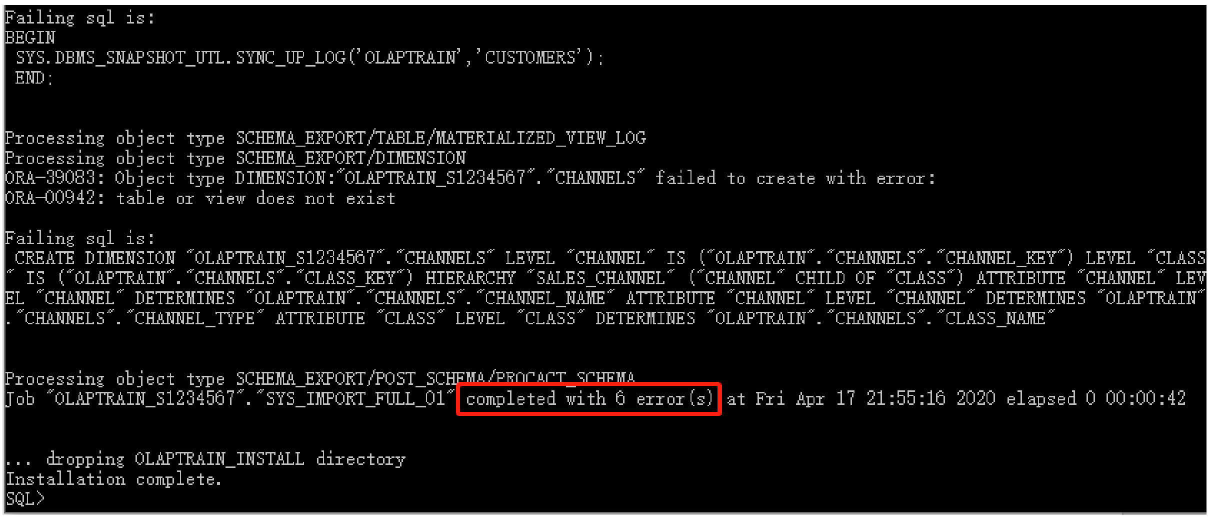
Note that the directory should be the same as above and you can choose your own password for user "OLAPTRAIN_S1234567" (Please **DO NOT** include '-' or other special characters in your password), which will be the main user throughout this practical. Please change "S1234567" to your student ID and make sure "S" is upper case, this is crucial. The processes are shown below:



```
c:\app\INFS3200\P2\olaptrain_install>sqlplus sys as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Fri Apr 17 21:39:34 2020

Copyright (c) 1982, 2016, Oracle.  All rights reserved.

Enter password:
Last Successful login time: Fri Apr 17 2020 21:34:44 +10:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

SQL> alter session set "_ORACLE_SCRIPT"=true;

Session altered.

SQL> @install_olaptrain_student

This procedure installs the OLAPTRAIN schema for the Oracle OLAP 12c
training class, samples and Oracle by Examples.

1. You should be logged on as a DBA to execute this procedure.
2. Ensure any existing OLAPTRAIN user sessions are disconnected before proceding with the installation (this install wil
l recreate the user if it exists)

Specify file system directory containing this installation program.

Default:
c:\app\P2\olaptrain_install

Directory: c:\app\P2\olaptrain_install

Specify a password for the OLAPTRAIN user.
Password:
please type in your student ID, e.g.:S1234567, and your final username will be OLAPTRAIN_S1234567
PLEASE MAKE SURE 'S' IS UPPER CASE.
Student ID: S1234567_
```

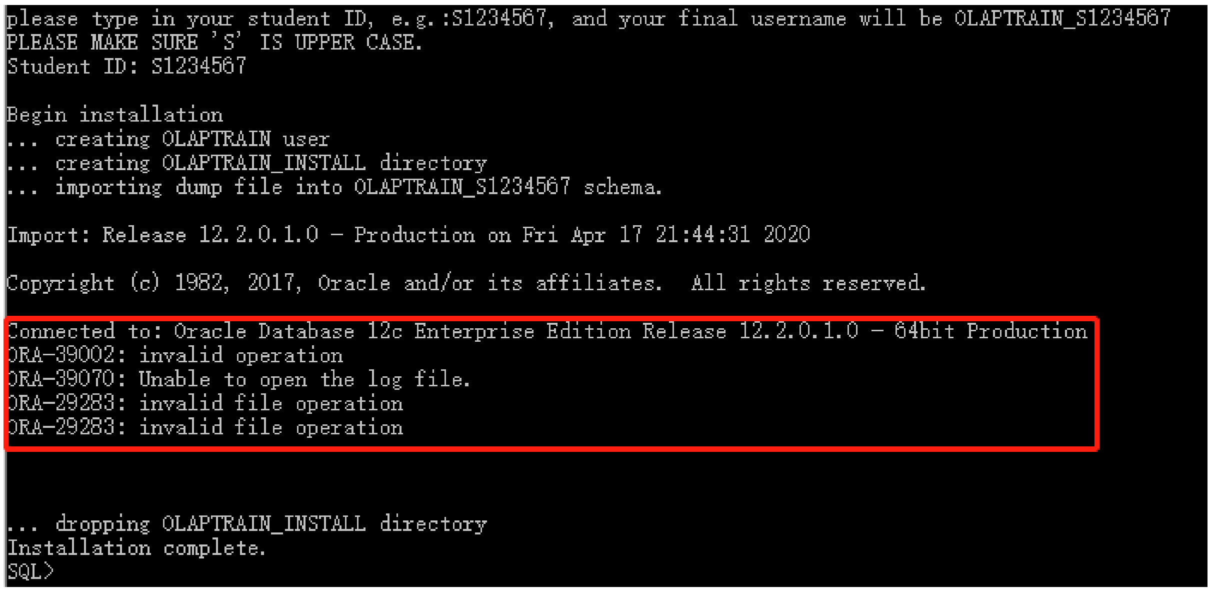
Troubleshooting:

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



```
Failing sql is:
BEGIN
 SYS.DBMS_SNAPSHOT_UTL.SYNC_UP_LOG('OLAPTRAIN','CUSTOMERS');
 END;

Processing object type SCHEMA_EXPORT/TABLE/MATERIALIZED_VIEW_LOG
Processing object type SCHEMA_EXPORT/DIMENSION
ORA-39083: Object type DIMENSION:"OLAPTRAIN_S1234567"."CHANNELS" failed to create with error:
ORA-00942: table or view does not exist

Failing sql is:
 CREATE DIMENSION "OLAPTRAIN_S1234567"."CHANNELS" LEVEL "CHANNEL" IS ("OLAPTRAIN"."CHANNELS"."CHANNEL_KEY") LEVEL "CLASS
" IS ("OLAPTRAIN"."CHANNELS"."CLASS_KEY") HIERARCHY "SALES_CHANNEL" ("CHANNEL" CHILD OF "CLASS") ATTRIBUTE "CHANNEL" LEV
EL "CHANNEL" DETERMINES "OLAPTRAIN"."CHANNELS"."CHANNEL_NAME" ATTRIBUTE "CHANNEL" LEVEL "CHANNEL" DETERMINES "OLAPTRAIN"
."CHANNELS"."CHANNEL_TYPE" ATTRIBUTE "CLASS" LEVEL "CLASS" DETERMINES "OLAPTRAIN"."CHANNELS"."CLASS_NAME"

Processing object type SCHEMA_EXPORT/POST_SCHEMA/PROCACT_SCHEMA
Job "OLAPTRAIN_S1234567"."SYS_IMPORT_FULL_01" completed with 6 error(s) at Fri Apr 17 21:55:16 2020 elapsed 0 00:00:42

... dropping OLAPTRAIN_INSTALL directory
Installation complete.
SQL>
```

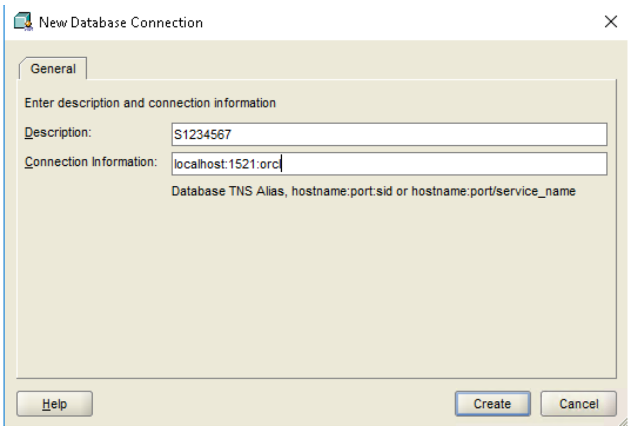
However, if the installation ends quickly with the following messages, it is usually caused by specifying an incorrect directory.



```
please type in your student ID, e.g.:S1234567, and your final username will be OLAPTRAIN_S1234567
PLEASE MAKE SURE 'S' IS UPPER CASE.
Student ID: S1234567

Begin installation
... creating OLAPTRAIN user
... creating OLAPTRAIN_INSTALL directory
... importing dump file into OLAPTRAIN_S1234567 schema.

Import: Release 12.2.0.1.0 - Production on Fri Apr 17 21:44:31 2020

Copyright (c) 1982, 2017, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
ORA-39002: invalid operation
ORA-39070: Unable to open the log file.
ORA-29283: invalid file operation
ORA-29283: invalid file operation


... dropping OLAPTRAIN_INSTALL directory
Installation complete.
SQL>
```

4. 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.

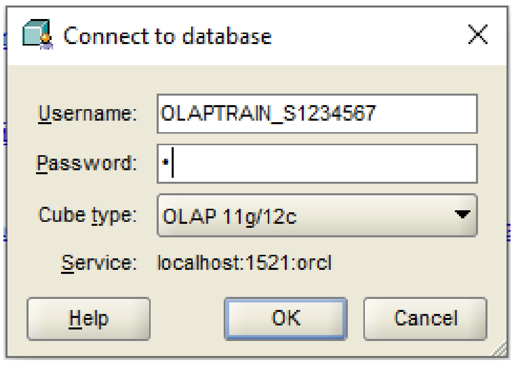


Enter description and connection information

Description: S1234567

Connection Information: localhost:1521:orcl

Database TNS Alias, hostname:port:sid or hostname:port/service_name

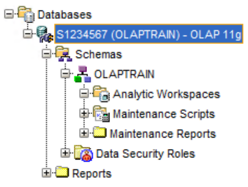


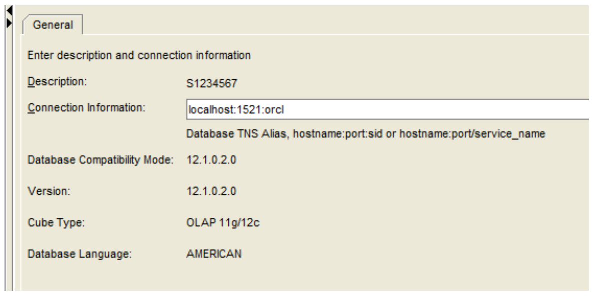
Username: OLAPTRAIN_S1234567

Password: •

Cube type: OLAP 11g/12c

Service: localhost:1521:orcl





Enter description and connection information

Description: S1234567

Connection Information: localhost:1521:orcl

Database TNS Alias, hostname:port:sid or hostname:port/service_name

Database Compatibility Mode: 12.1.0.2.0

Version: 12.1.0.2.0

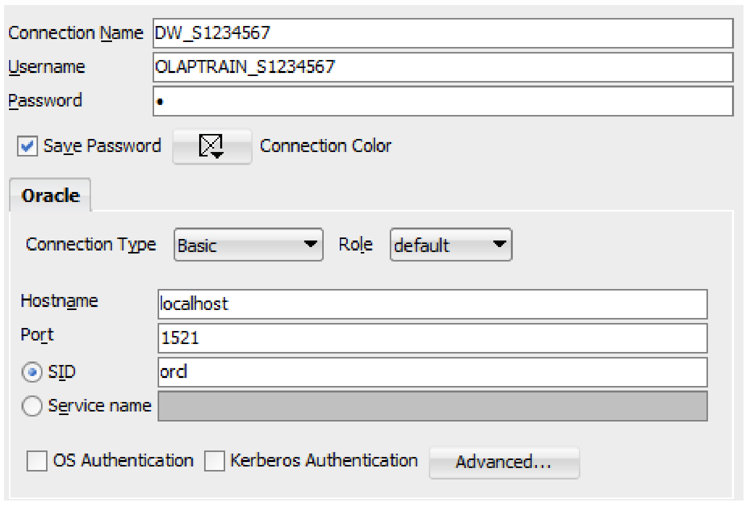
Cube Type: OLAP 11g/12c

Database Language: AMERICAN

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”.



Connection Name: DW_S1234567

Username: OLAPTRAIN_S1234567

Password: •

[✓] Save Password

Connection Type: Basic   Role: default

Hostname: localhost

Port: 1521

[● SID]: orcl

Service name:

[ ] OS Authentication [ ] Kerberos Authentication

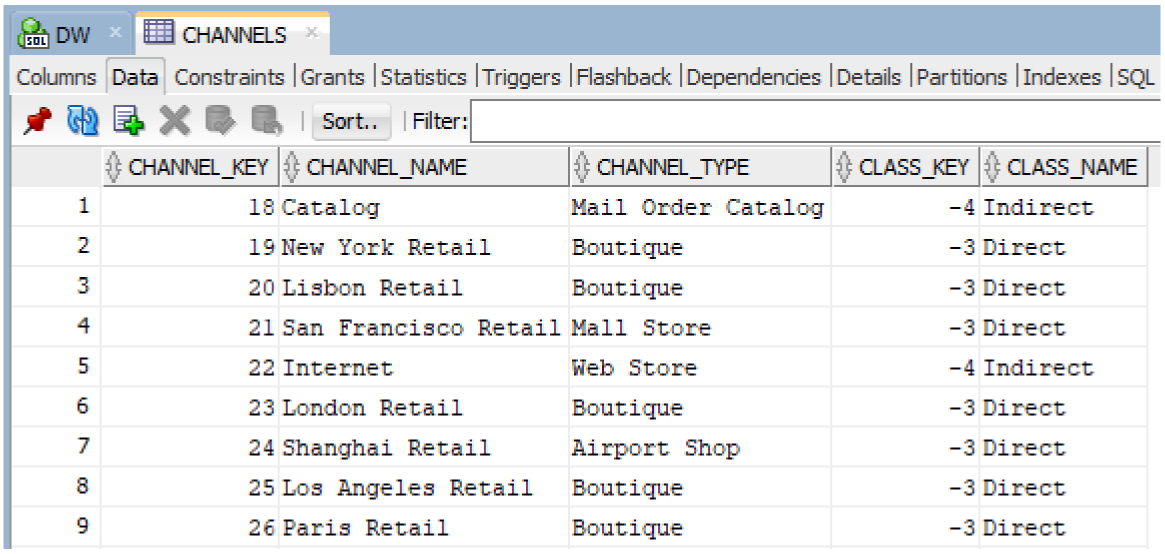
After examining the OLAPTRAIN schema, we need to identify the dimensions, hierarchies and attributes of our data model. In addition, we define various measures based on business interests.

(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).


DW
CHANNELS
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

	CHANNEL_KEY	CHANNEL_NAME	CHANNEL_TYPE	CLASS_KEY	CLASS_NAME
1	18	Catalog	Mail Order Catalog	-4	Indirect
2	19	New York Retail	Boutique	-3	Direct
3	20	Lisbon Retail	Boutique	-3	Direct
4	21	San Francisco Retail	Mall Store	-3	Direct
5	22	Internet	Web Store	-4	Indirect
6	23	London Retail	Boutique	-3	Direct
7	24	Shanghai Retail	Airport Shop	-3	Direct
8	25	Los Angeles Retail	Boutique	-3	Direct
9	26	Paris Retail	Boutique	-3	Direct

(2) 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:

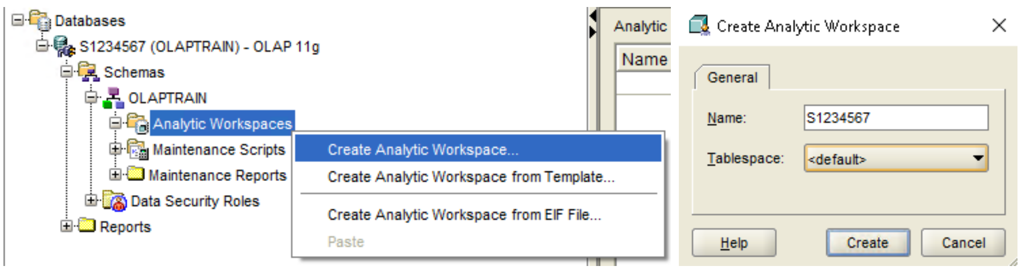
Stored Measures
- Sales
- Quantity

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.

2. 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.

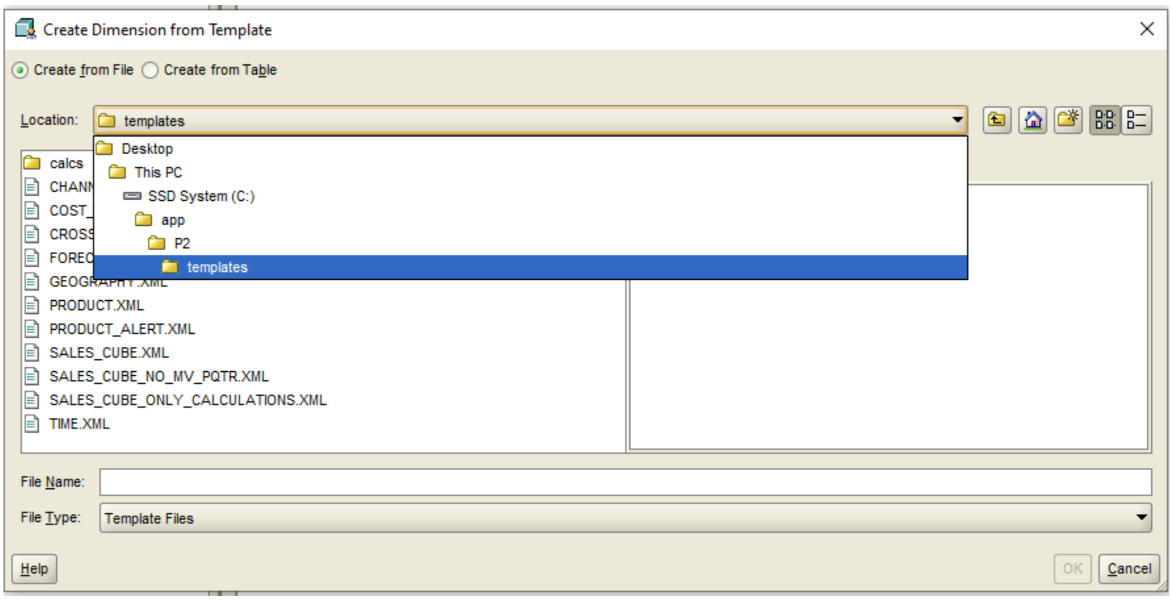

Databases
S1234567 (OLAPTRAIN) - OLAP 11g
Schemas
OLAPTRAIN
Analytic Workspaces
Maintenance Scripts
Maintenance Reports
Data Security Roles
Reports
Create Analytic Workspace...
Create Analytic Workspace from Template...
Create Analytic Workspace from EIF File...
Paste
Create Analytic Workspace
General
Help
Create
Cancel

Name: S1234567
Tablespace: <default>

3. 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.

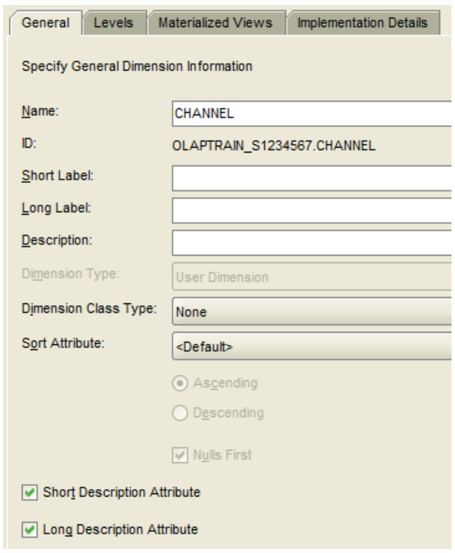
In order to import a template, right-click the **Dimensions** folder, then select **Create Dimension from Template.** The templates are stored in the *templates* folder in the extracted folder.



Create from File [✓]
Location: templates
File Name:
File Type: Template Files

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).



Specify General Dimension Information

Name: CHANNEL

ID: OLAPTRAIN_S1234567.CHANNEL

Short Label:

Long Label:

Description:

Dimension Type: User Dimension

Dimension Class Type: None

Sort Attribute: <Default>

Ascending

Descending

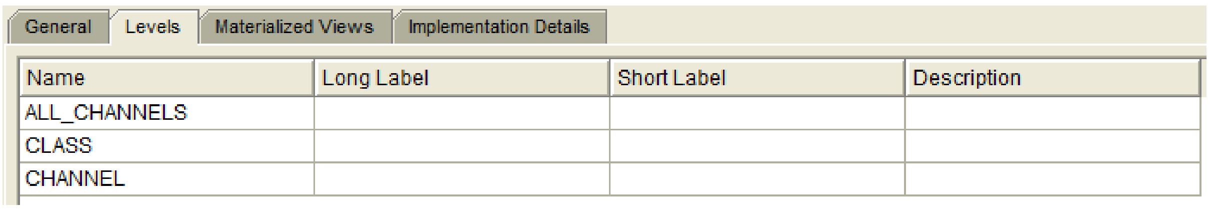
✓ Nulls First

✓ Short Description Attribute

✓ Long Description Attribute

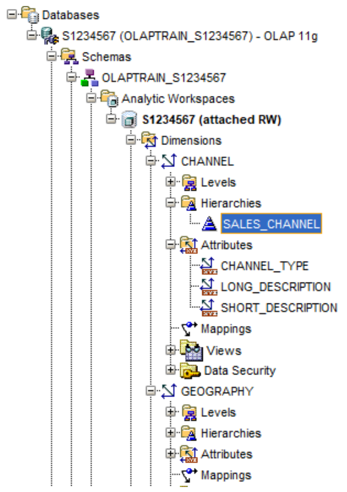
In the Levels tab, we can see three levels are defined, which correspond to three of the four possible dimensions mentioned above (channel type not included):

- **ALL_CHANNELS** = Ø
- **CLASS** = channel class
- **CHANNEL** = the actual channels

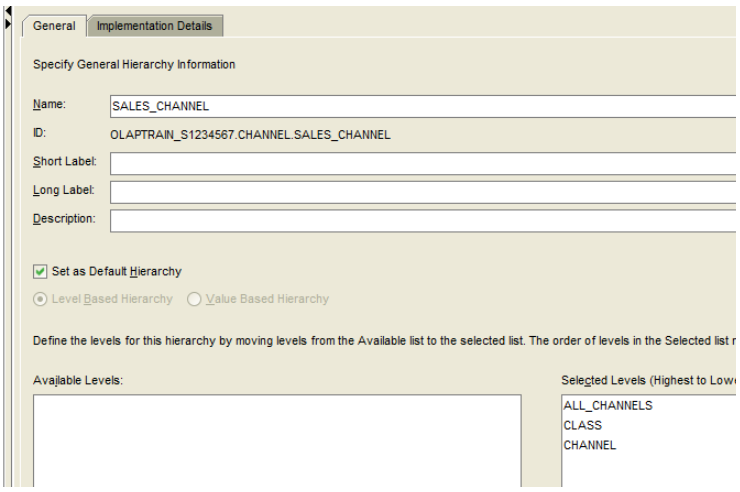


Name	Long Label	Short Label	Description
ALL_CHANNELS			
CLASS			
CHANNEL			

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:



```
Databases
  S1234567 (OLAPTRAIN_S1234567) - OLAP 11g
    Schemas
      OLAPTRAIN_S1234567
        Analytic Workspaces
          S1234567 (attached RW)
            Dimensions
              CHANNEL
                Levels
                Hierarchies
                  SALES_CHANNEL
                Attributes
                  CHANNEL_TYPE
                  LONG_DESCRIPTION
                  SHORT_DESCRIPTION
                Mappings
                Views
                Data Security
              GEOGRAPHY
                Levels
                Hierarchies
                Attributes
                Mappings
```



Specify General Hierarchy Information

Name: SALES_CHANNEL

ID: OLAPTRAIN_S1234567.CHANNEL.SALES_CHANNEL

Short Label:

Long Label:

Description:

✓ Set as Default Hierarchy

Level Based Hierarchy   Value Based Hierarchy

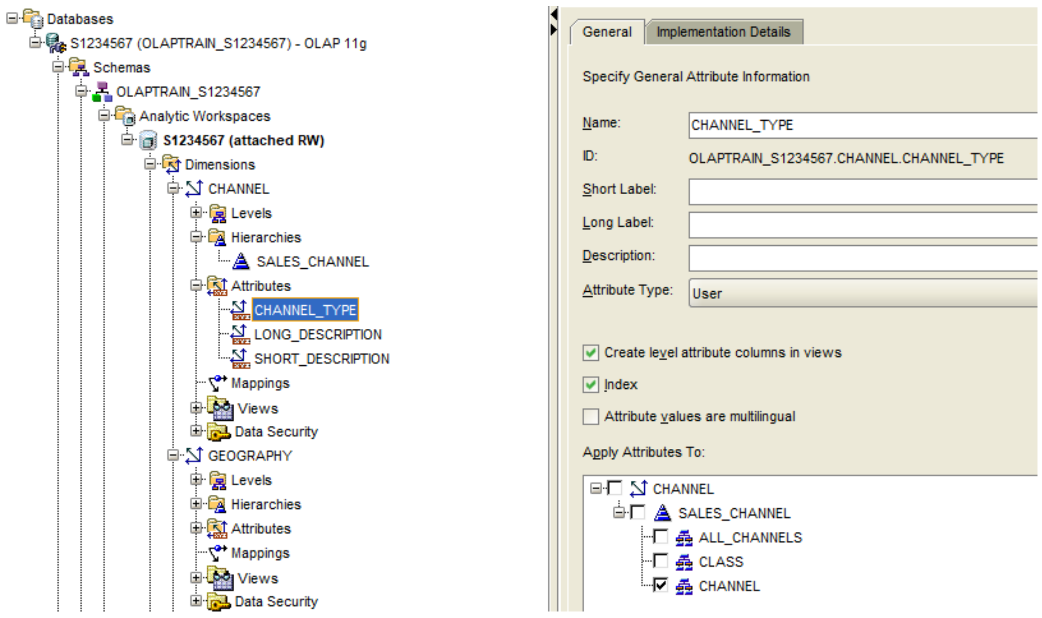
Define the levels for this hierarchy by moving levels from the Available list to the selected list. The order of levels in the Selected list r

Available Levels:

Selected Levels (Highest to Low

ALL_CHANNELS
CLASS
CHANNEL

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.



```
Databases
  S1234567 (OLAPTRAIN_S1234567) - OLAP 11g
    Schemas
      OLAPTRAIN_S1234567
        Analytic Workspaces
          S1234567 (attached RW)
            Dimensions
              CHANNEL
                Levels
                Hierarchies
                  SALES_CHANNEL
                Attributes
                  CHANNEL_TYPE
                  LONG_DESCRIPTION
                  SHORT_DESCRIPTION
                Mappings
                Views
                Data Security
              GEOGRAPHY
                Levels
                Hierarchies
                Attributes
                Mappings
                Views
                Data Security
```

General | Implementation Details

Specify General Attribute Information

Name:	CHANNEL_TYPE
ID:	OLAPTRAIN_S1234567.CHANNEL.CHANNEL_TYPE
Short Label:	
Long Label:	
Description:	
Attribute Type:	User

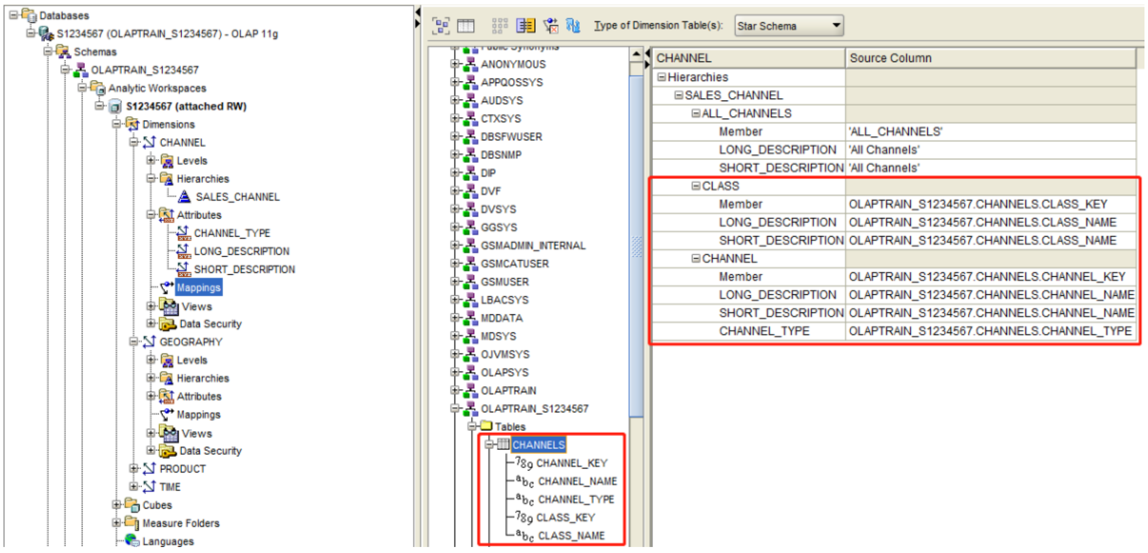
[✓] Create level attribute columns in views
[✓] Index
[ ] Attribute values are multilingual

Apply Attributes To:

```
[ ] CHANNEL
  [ ] SALES_CHANNEL
    [ ] ALL_CHANNELS
    [ ] CLASS
    [✓] CHANNEL
```

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:



```
Databases
  S1234567 (OLAPTRAIN_S1234567) - OLAP 11g
    Schemas
      OLAPTRAIN_S1234567
        Analytic Workspaces
          S1234567 (attached RW)
            Dimensions
              CHANNEL
                Levels
                Hierarchies
                  SALES_CHANNEL
                Attributes
                  CHANNEL_TYPE
                  LONG_DESCRIPTION
                  SHORT_DESCRIPTION
                Mappings
                Views
                Data Security
              GEOGRAPHY
                Levels
                Hierarchies
                Attributes
                Mappings
                Views
                Data Security
              PRODUCT
              TIME
            Cubes
            Measure Folders
            Languages
```

Type of Dimension Table(s): Star Schema

```
ANONYMOUS
APPQOSSYS
AUDSYS
CTXSYS
DBSFWUSER
DBSNMP
DIP
DVF
DVSYS
GGSYS
GSMADMIN_INTERNAL
GSMCATUSER
GSMUSER
LBACSYS
MDDATA
MDSYS
OJVMSYS
OLAPSYS
OLAPTRAIN
OLAPTRAIN_S1234567
  Tables
    CHANNELS
      CHANNEL_KEY
      CHANNEL_NAME
      CHANNEL_TYPE
      CLASS_KEY
      CLASS_NAME
```

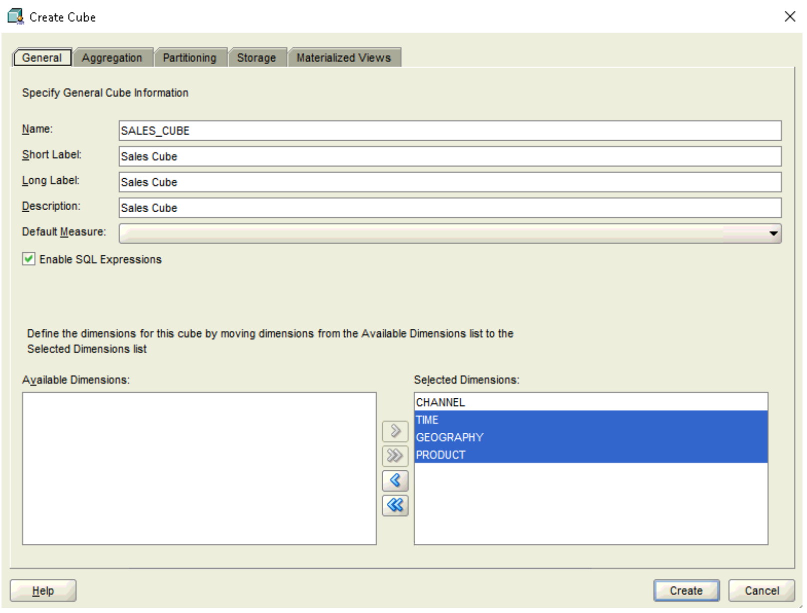
CHANNEL	Source Column
Hierarchies	
SALES_CHANNEL	
ALL_CHANNELS	
Member	'ALL_CHANNELS'
LONG_DESCRIPTION	'All Channels'
SHORT_DESCRIPTION	'All Channels'
CLASS	
Member	OLAPTRAIN_S1234567.CHANNELS.CLASS_KEY
LONG_DESCRIPTION	OLAPTRAIN_S1234567.CHANNELS.CLASS_NAME
SHORT_DESCRIPTION	OLAPTRAIN_S1234567.CHANNELS.CLASS_NAME
CHANNEL	
Member	OLAPTRAIN_S1234567.CHANNELS.CHANNEL_KEY
LONG_DESCRIPTION	OLAPTRAIN_S1234567.CHANNELS.CHANNEL_NAME
SHORT_DESCRIPTION	OLAPTRAIN_S1234567.CHANNELS.CHANNEL_NAME
CHANNEL_TYPE	OLAPTRAIN_S1234567.CHANNELS.CHANNEL_TYPE

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

4. 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



Create Cube

General | Aggregation | Partitioning | Storage | Materialized Views

Specify General Cube Information

Name:	SALES_CUBE
Short Label:	Sales Cube
Long Label:	Sales Cube
Description:	Sales Cube
Default Measure:	

☑ Enable SQL Expressions

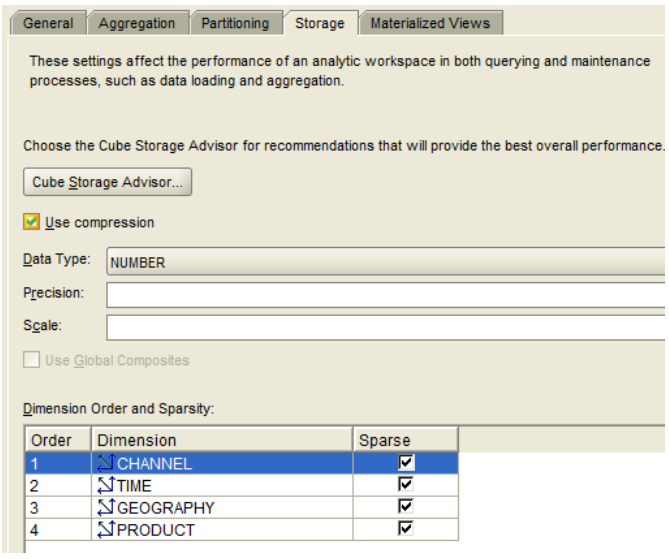
Define the dimensions for this cube by moving dimensions from the Available Dimensions list to the Selected Dimensions list

Available Dimensions:

Selected Dimensions:
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:



General | Aggregation | Partitioning | Storage | Materialized Views

These settings affect the performance of an analytic workspace in both querying and maintenance processes, such as data loading and aggregation.

Choose the Cube Storage Advisor for recommendations that will provide the best overall performance.

☑ Use compression

Data Type:	NUMBER
Precision:	
Scale:	

☐ Use Global Composites

Dimension Order and Sparsity:

Order	Dimension	Sparse
1	CHANNEL	✓
2	TIME	✓
3	GEOGRAPHY	✓
4	PRODUCT	✓

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

5. 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

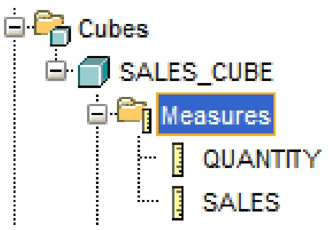
Stored measures store the facts collected about your business. When you create stored measures in your OLAP data model, you will map them to the source data just like what you have done with dimensions.

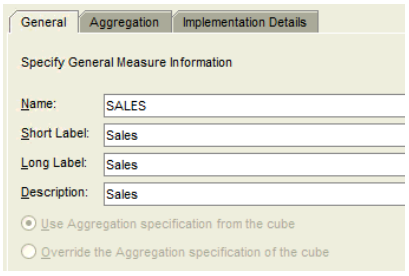
Calculated Measures

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:

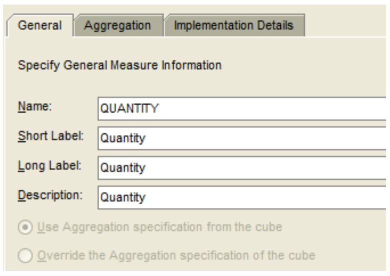




Specify General Measure Information

Name:	SALES
Short Label:	Sales
Long Label:	Sales
Description:	Sales

[✓ Use Aggregation specification from the cube]
Override the Aggregation specification of the cube



Specify General Measure Information

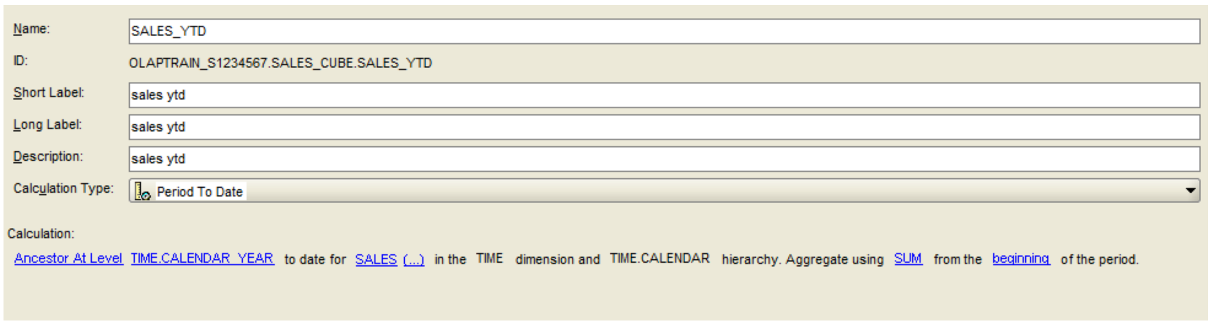
Name:	QUANTITY
Short Label:	Quantity
Long Label:	Quantity
Description:	Quantity

[✓ Use Aggregation specification from the cube]
Override the Aggregation specification of the cube

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:



Name: SALES_YTD

ID: OLAPTRAIN_S1234567.SALES_CUBE.SALES_YTD

Short Label: sales ytd

Long Label: sales ytd

Description: sales ytd

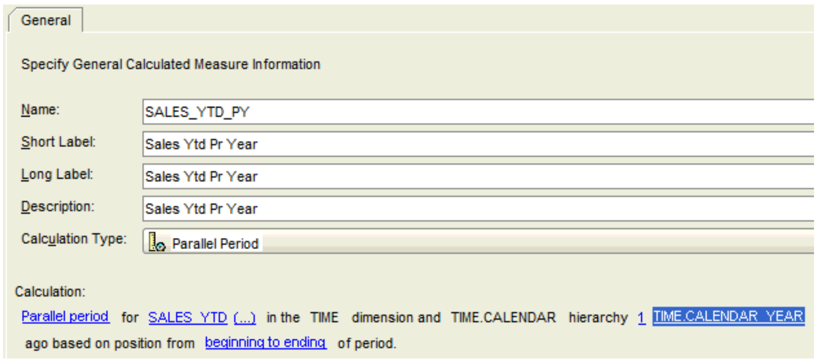
Calculation Type: Period To Date

Calculation:

Ancestor At Level TIME.CALENDAR_YEAR to date for SALES (...) in the TIME dimension and TIME.CALENDAR hierarchy. Aggregate using SUM from the beginning of the period.

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



Specify General Calculated Measure Information

Name: SALES_YTD_PY

Short Label: Sales Ytd Pr Year

Long Label: Sales Ytd Pr Year

Description: Sales Ytd Pr Year

Calculation Type: Parallel Period

Calculation:

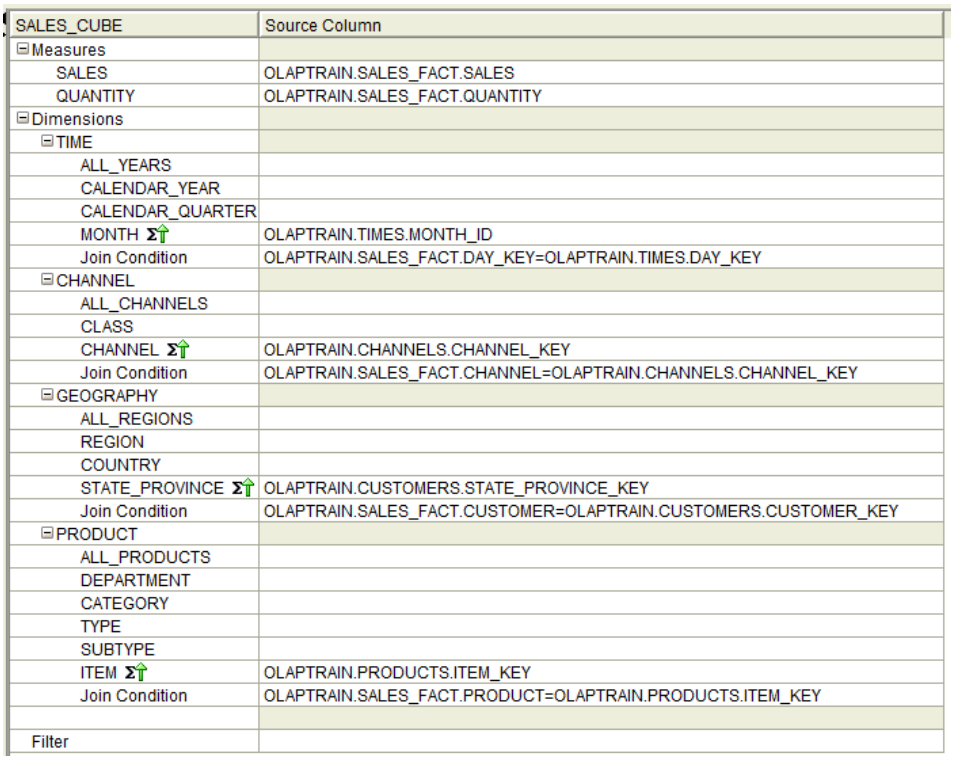
Parallel period for SALES_YTD (...) in the TIME dimension and TIME.CALENDAR hierarchy 1 TIME.CALENDAR_YEAR ago based on position from beginning to ending of period.

6. 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.

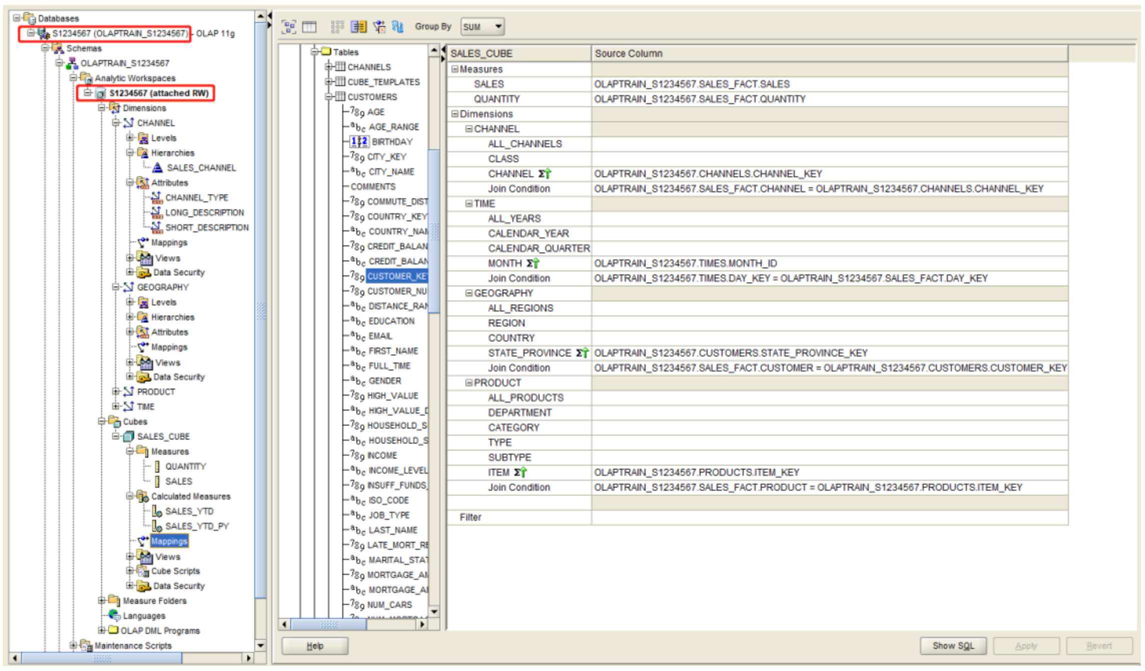
Note that the mapping is done by dragging the corresponding column from the *source columns* (left) to the correct spot in the *mapping pane* (right). A Join condition can be achieved by dragging both joining columns to that slot and the “=” will be added automatically. Please DO NOT type those values 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.



SALES_CUBE	Source Column
⊟Measures	
SALES	OLAPTRAIN.SALES_FACT.SALES
QUANTITY	OLAPTRAIN.SALES_FACT.QUANTITY
⊟Dimensions	
⊟TIME	
ALL_YEARS	
CALENDAR_YEAR	
CALENDAR_QUARTER	
MONTH Σ	OLAPTRAIN.TIMES.MONTH_ID
Join Condition	OLAPTRAIN.SALES_FACT.DAY_KEY=OLAPTRAIN.TIMES.DAY_KEY
⊟CHANNEL	
ALL_CHANNELS	
CLASS	
CHANNEL Σ	OLAPTRAIN.CHANNELS.CHANNEL_KEY
Join Condition	OLAPTRAIN.SALES_FACT.CHANNEL=OLAPTRAIN.CHANNELS.CHANNEL_KEY
⊟GEOGRAPHY	
ALL_REGIONS	
REGION	
COUNTRY	
STATE_PROVINCE Σ	OLAPTRAIN.CUSTOMERS.STATE_PROVINCE_KEY
Join Condition	OLAPTRAIN.SALES_FACT.CUSTOMER=OLAPTRAIN.CUSTOMERS.CUSTOMER_KEY
⊟PRODUCT	
ALL_PRODUCTS	
DEPARTMENT	
CATEGORY	
TYPE	
SUBTYPE	
ITEM Σ	OLAPTRAIN.PRODUCTS.ITEM_KEY
Join Condition	OLAPTRAIN.SALES_FACT.PRODUCT=OLAPTRAIN.PRODUCTS.ITEM_KEY
Filter	

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

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:

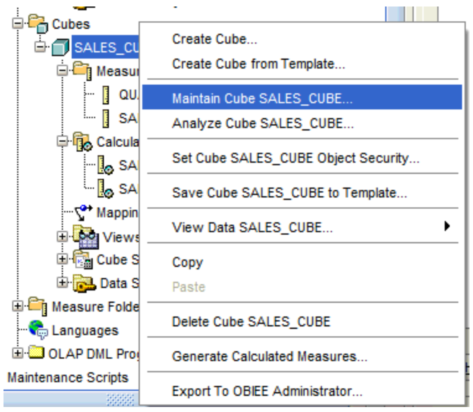


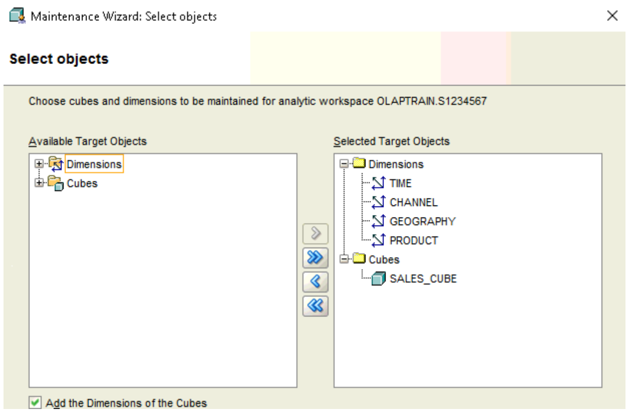
SALES_CUBE	Source Column
⊟Measures	
SALES	OLAPTRAIN_S1234567.SALES_FACT.SALES
QUANTITY	OLAPTRAIN_S1234567.SALES_FACT.QUANTITY
⊟Dimensions	
⊟CHANNEL	
ALL_CHANNELS	
CLASS	
CHANNEL Σ	OLAPTRAIN_S1234567.CHANNELS.CHANNEL_KEY
Join Condition	OLAPTRAIN_S1234567.SALES_FACT.CHANNEL = OLAPTRAIN_S1234567.CHANNELS.CHANNEL_KEY
⊟TIME	
ALL_YEARS	
CALENDAR_YEAR	
CALENDAR_QUARTER	
MONTH Σ	OLAPTRAIN_S1234567.TIMES.MONTH_ID
Join Condition	OLAPTRAIN_S1234567.TIMES.DAY_KEY = OLAPTRAIN_S1234567.SALES_FACT.DAY_KEY
⊟GEOGRAPHY	
ALL_REGIONS	
REGION	
COUNTRY	
STATE_PROVINCE Σ	OLAPTRAIN_S1234567.CUSTOMERS.STATE_PROVINCE_KEY
Join Condition	OLAPTRAIN_S1234567.SALES_FACT.CUSTOMER = OLAPTRAIN_S1234567.CUSTOMERS.CUSTOMER_KEY
⊟PRODUCT	
ALL_PRODUCTS	
DEPARTMENT	
CATEGORY	
TYPE	
SUBTYPE	
ITEM Σ	OLAPTRAIN_S1234567.PRODUCTS.ITEM_KEY
Join Condition	OLAPTRAIN_S1234567.SALES_FACT.PRODUCT = OLAPTRAIN_S1234567.PRODUCTS.ITEM_KEY
Filter	

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

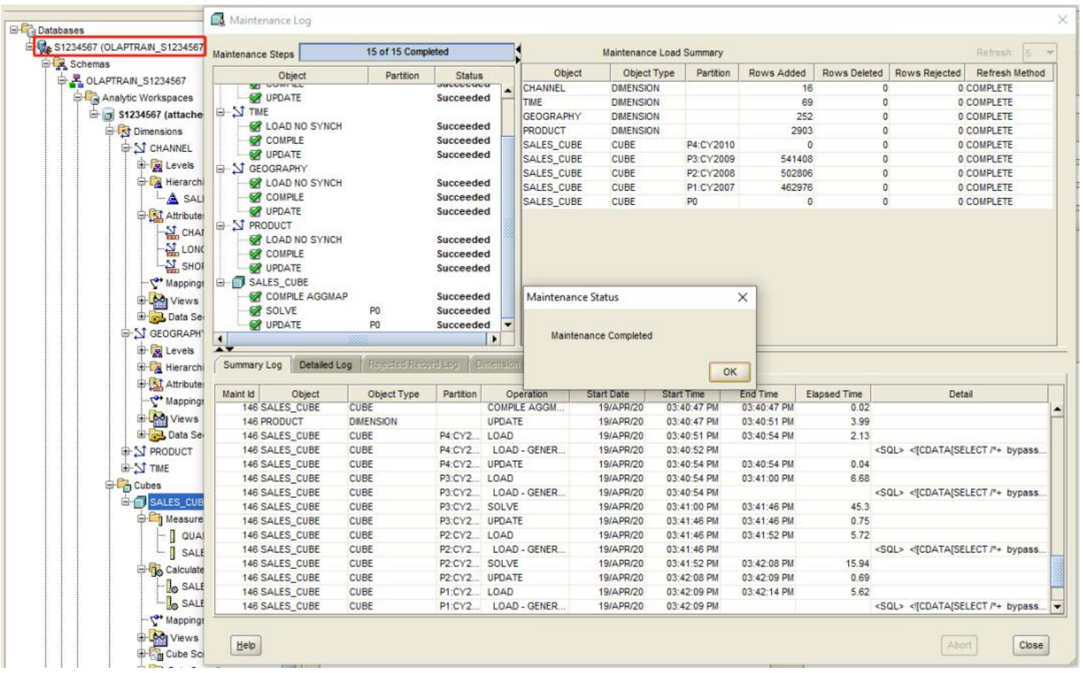

Cubes
SALES_CU
Measu
QU
SA
Calcula
SA
SA
Mappin
Views
Cube S
Data S
Measure Folde
Languages
OLAP DML Pro
Maintenance Scripts
Create Cube...
Create Cube from Template...
Maintain Cube SALES_CUBE...
Analyze Cube SALES_CUBE...
Set Cube SALES_CUBE Object Security...
Save Cube SALES_CUBE to Template...
View Data SALES_CUBE...
Copy
Paste
Delete Cube SALES_CUBE
Generate Calculated Measures...
Export To OBIEE Administrator...



Choose cubes and dimensions to be maintained for analytic workspace OLAPTRAIN.S1234567

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.*



Object	Partition	Status
UPDATE		Succeeded
TIME		
LOAD NO SYNCH		Succeeded
COMPILE		Succeeded
UPDATE		Succeeded
GEOGRAPHY		
LOAD NO SYNCH		Succeeded
COMPILE		Succeeded
UPDATE		Succeeded
PRODUCT		
LOAD NO SYNCH		Succeeded
COMPILE		Succeeded
UPDATE		Succeeded
SALES_CUBE		
COMPILE AGGMAP		Succeeded
SOLVE	P0	Succeeded
UPDATE	P0	Succeeded

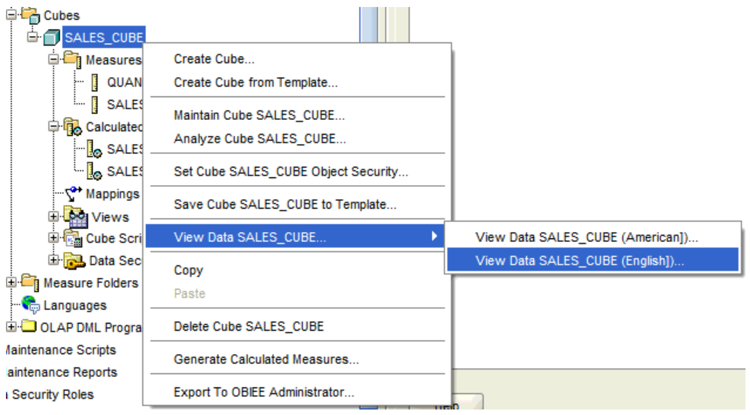
Maintenance Load Summary

Object	Object Type	Partition	Rows Added	Rows Deleted	Rows Rejected	Refresh Method
CHANNEL	DIMENSION		16	0	0	COMPLETE
TIME	DIMENSION		69	0	0	COMPLETE
GEOGRAPHY	DIMENSION		252	0	0	COMPLETE
PRODUCT	DIMENSION		2903	0	0	COMPLETE
SALES_CUBE	CUBE	P4:CY2010	0	0	0	COMPLETE
SALES_CUBE	CUBE	P3:CY2009	541408	0	0	COMPLETE
SALES_CUBE	CUBE	P2:CY2008	502806	0	0	COMPLETE
SALES_CUBE	CUBE	P1:CY2007	462976	0	0	COMPLETE
SALES_CUBE	CUBE	P0	0	0	0	COMPLETE

Maint Id	Object	Object Type	Partition	Operation	Start Date	Start Time	End Time	Elapsed Time	Detail
146	SALES_CUBE	CUBE		COMPILE AGGM...	19/APR/20	03:40:47 PM	03:40:47 PM	0.02	
146	PRODUCT	DIMENSION		UPDATE	19/APR/20	03:40:47 PM	03:40:51 PM	3.99	
146	SALES_CUBE	CUBE	P4:CY2...	LOAD	19/APR/20	03:40:51 PM	03:40:54 PM	2.13	
146	SALES_CUBE	CUBE	P4:CY2...	LOAD - GENER...	19/APR/20	03:40:52 PM			<SQL> <![CDATA[SELECT /*+ bypass...
146	SALES_CUBE	CUBE	P4:CY2...	UPDATE	19/APR/20	03:40:54 PM	03:40:54 PM	0.04	
146	SALES_CUBE	CUBE	P3:CY2...	LOAD	19/APR/20	03:40:54 PM	03:41:00 PM	6.68	
146	SALES_CUBE	CUBE	P3:CY2...	LOAD - GENER...	19/APR/20	03:40:54 PM			<SQL> <![CDATA[SELECT /*+ bypass...
146	SALES_CUBE	CUBE	P3:CY2...	SOLVE	19/APR/20	03:41:00 PM	03:41:46 PM	45.3	
146	SALES_CUBE	CUBE	P3:CY2...	UPDATE	19/APR/20	03:41:46 PM	03:41:46 PM	0.75	
146	SALES_CUBE	CUBE	P2:CY2...	LOAD	19/APR/20	03:41:46 PM	03:41:52 PM	5.72	
146	SALES_CUBE	CUBE	P2:CY2...	LOAD - GENER...	19/APR/20	03:41:46 PM			<SQL> <![CDATA[SELECT /*+ bypass...
146	SALES_CUBE	CUBE	P2:CY2...	SOLVE	19/APR/20	03:41:52 PM	03:42:08 PM	15.94	
146	SALES_CUBE	CUBE	P2:CY2...	UPDATE	19/APR/20	03:42:08 PM	03:42:09 PM	0.69	
146	SALES_CUBE	CUBE	P1:CY2...	LOAD	19/APR/20	03:42:09 PM	03:42:14 PM	5.62	
146	SALES_CUBE	CUBE	P1:CY2...	LOAD - GENER...	19/APR/20	03:42:09 PM			<SQL> <![CDATA[SELECT /*+ bypass...

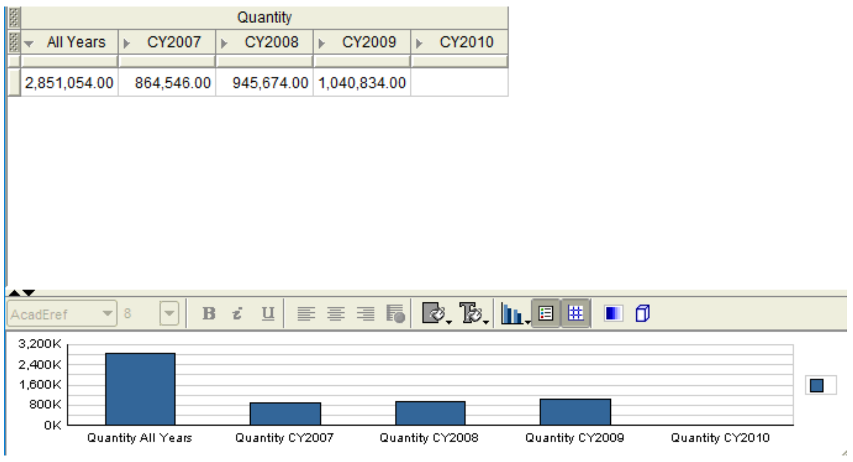
2. View the cube data

Click the **View Data SALES_CUBE** to start the Data Viewer.

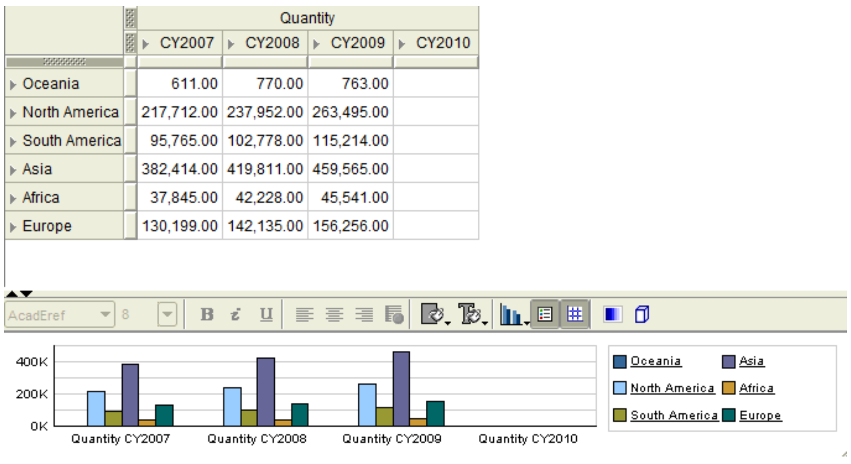

Cubes
SALES_CUBE
Measures
QUAN
SALES
Calculated
SALES
SALES
Mappings
Views
Cube Scri
Data Sec
Measure Folders
Languages
OLAP DML Progra
laintenance Scripts
aintenance Reports
Security Roles
Create Cube...
Create Cube from Template...
Maintain Cube SALES_CUBE...
Analyze Cube SALES_CUBE...
Set Cube SALES_CUBE Object Security...
Save Cube SALES_CUBE to Template...
View Data SALES_CUBE...
View Data SALES_CUBE (American])...
View Data SALES_CUBE (English])...
Copy
Paste
Delete Cube SALES_CUBE
Generate Calculated Measures...
Export To OBIEE Administrator...

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

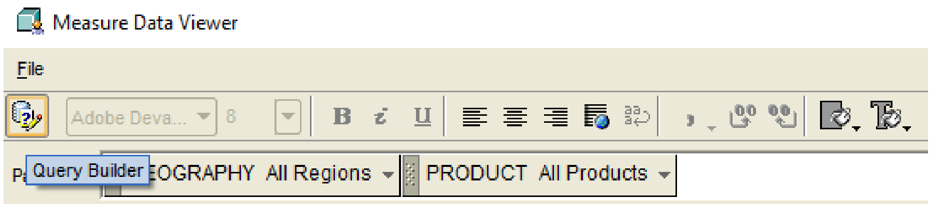
- Perform roll-up, drill-down and pivot operations in Data Viewer, respectively (Lecture5-P36, Tutorial4-Q2). Each operation includes three parts: (1) the screenshot of the view before the operation, (2) a sentence describing which operation you are performing on which columns/levels; (3) the screenshot 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.*



Quantity				
All Years	CY2007	CY2008	CY2009	CY2010
2,851,054.00	864,546.00	945,674.00	1,040,834.00	



	Quantity			
	CY2007	CY2008	CY2009	CY2010
Oceania	611.00	770.00	763.00	
North America	217,712.00	237,952.00	263,495.00	
South America	95,765.00	102,778.00	115,214.00	
Asia	382,414.00	419,811.00	459,565.00	
Africa	37,845.00	42,228.00	45,541.00	
Europe	130,199.00	142,135.00	156,256.00	

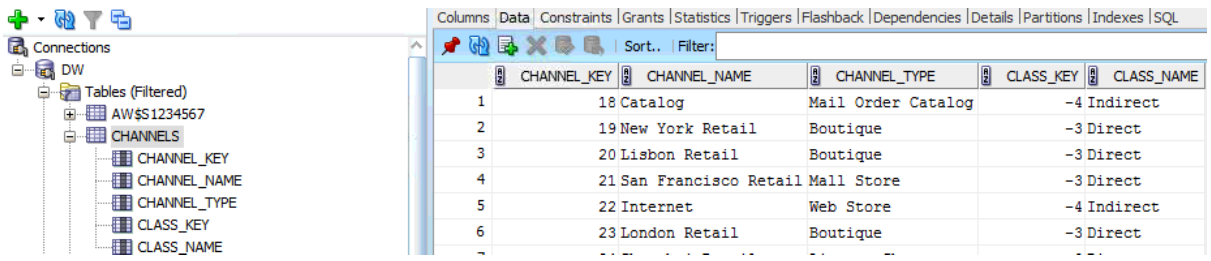


Part 3: Understand Data Warehouse Design Mechanism

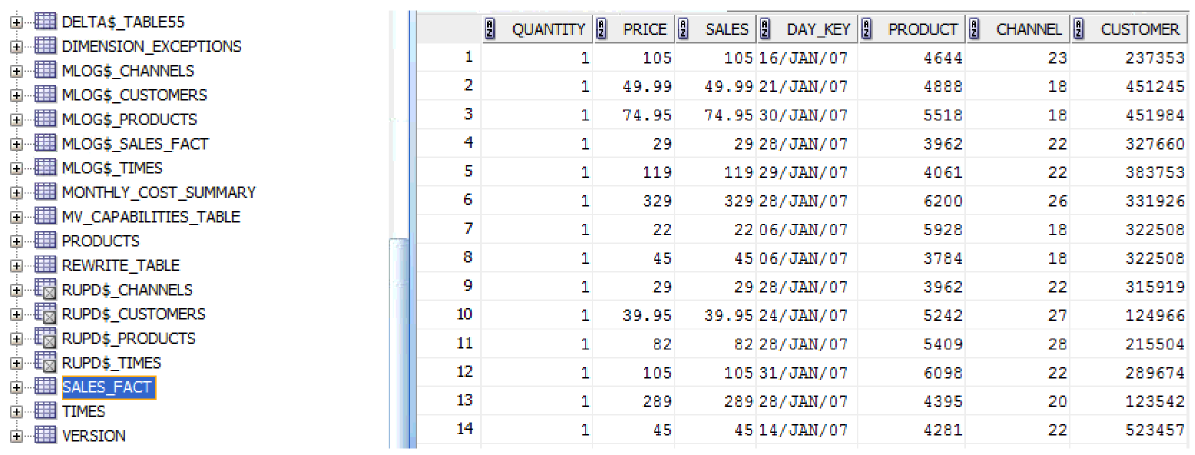
We have learnt how to create dimensions and cubes using Oracle OLAP and AWM. Now, we will try to understand the mechanism behind. Specifically, there is **NO mark** in this part, but it will help you understand the whole process of how data cubes are built, queried and managed by Oracle OLAP, which will 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".



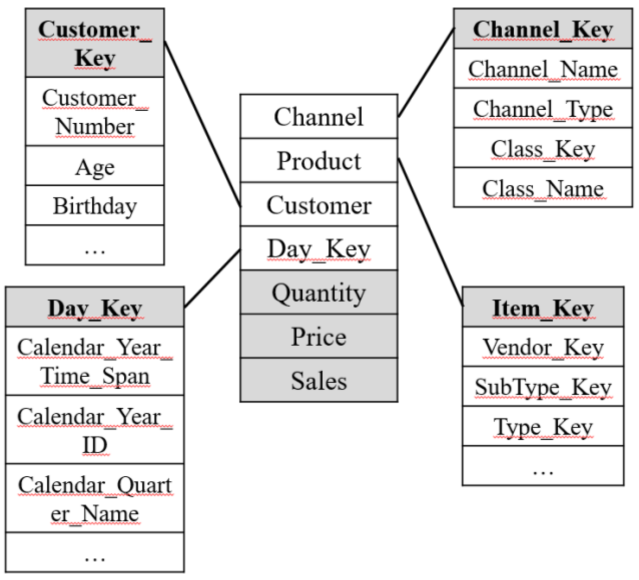
	CHANNEL_KEY	CHANNEL_NAME	CHANNEL_TYPE	CLASS_KEY	CLASS_NAME
1	18	Catalog	Mail Order Catalog	-4	Indirect
2	19	New York Retail	Boutique	-3	Direct
3	20	Lisbon Retail	Boutique	-3	Direct
4	21	San Francisco Retail	Mall Store	-3	Direct
5	22	Internet	Web Store	-4	Indirect
6	23	London Retail	Boutique	-3	Direct



- DELTA$_TABLE55
- DIMENSION_EXCEPTIONS
- MLOG$_CHANNELS
- MLOG$_CUSTOMERS
- MLOG$_PRODUCTS
- MLOG$_SALES_FACT
- MLOG$_TIMES
- MONTHLY_COST_SUMMARY
- MV_CAPABILITIES_TABLE
- PRODUCTS
- REWRITE_TABLE
- RUPD$_CHANNELS
- RUPD$_CUSTOMERS
- RUPD$_PRODUCTS
- RUPD$_TIMES
- SALES_FACT
- TIMES
- VERSION

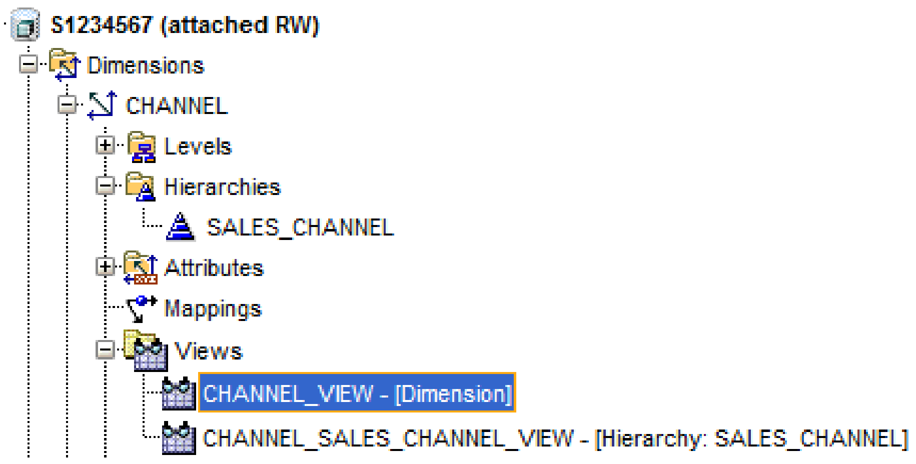
	QUANTITY	PRICE	SALES	DAY_KEY	PRODUCT	CHANNEL	CUSTOMER
1	1	105	105	16/JAN/07	4644	23	237353
2	1	49.99	49.99	21/JAN/07	4888	18	451245
3	1	74.95	74.95	30/JAN/07	5518	18	451984
4	1	29	29	28/JAN/07	3962	22	327660
5	1	119	119	29/JAN/07	4061	22	383753
6	1	329	329	28/JAN/07	6200	26	331926
7	1	22	22	06/JAN/07	5928	18	322508
8	1	45	45	06/JAN/07	3784	18	322508
9	1	29	29	28/JAN/07	3962	22	315919
10	1	39.95	39.95	24/JAN/07	5242	27	124966
11	1	82	82	28/JAN/07	5409	28	215504
12	1	105	105	31/JAN/07	6098	22	289674
13	1	289	289	28/JAN/07	4395	20	123542
14	1	45	45	14/JAN/07	4281	22	523457

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:



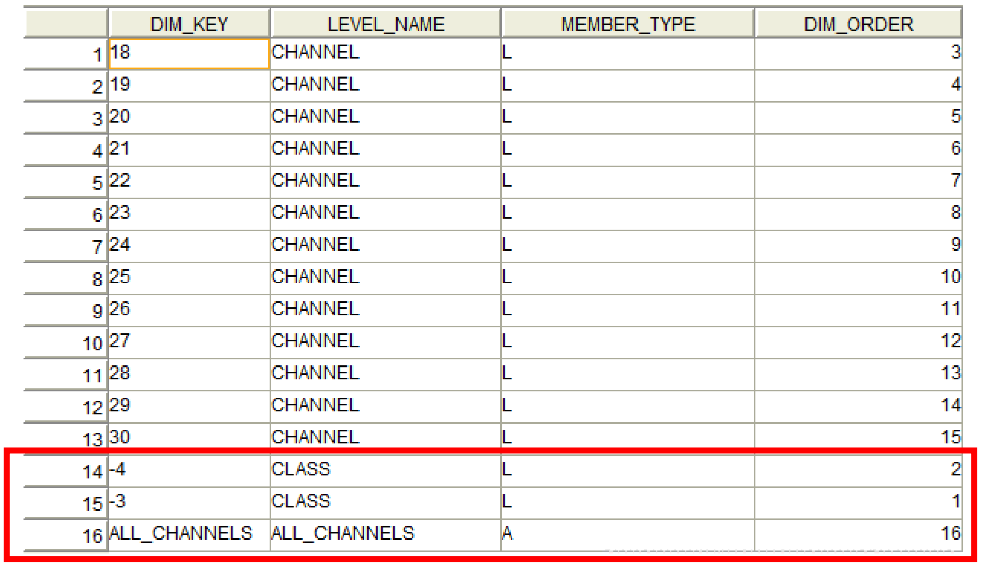
2. Compare the views with dimension tables and fact table

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".

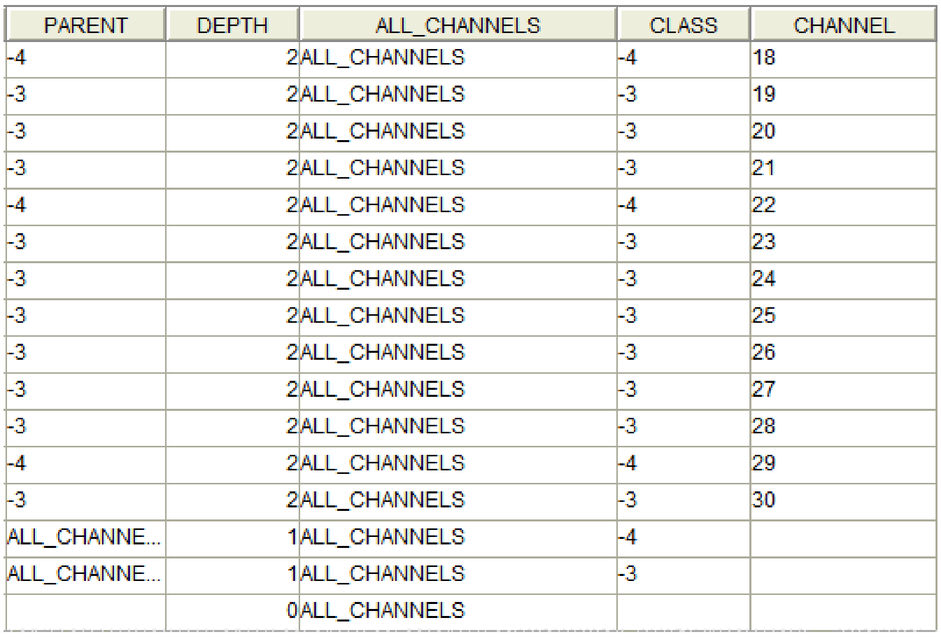


- S1234567 (attached RW)
  - Dimensions
    - CHANNEL
      - Levels
      - Hierarchies
        - SALES_CHANNEL
      - Attributes
      - Mappings
      - Views
        - CHANNEL_VIEW - [Dimension]
        - CHANNEL_SALES_CHANNEL_VIEW - [Hierarchy: SALES_CHANNEL]

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.

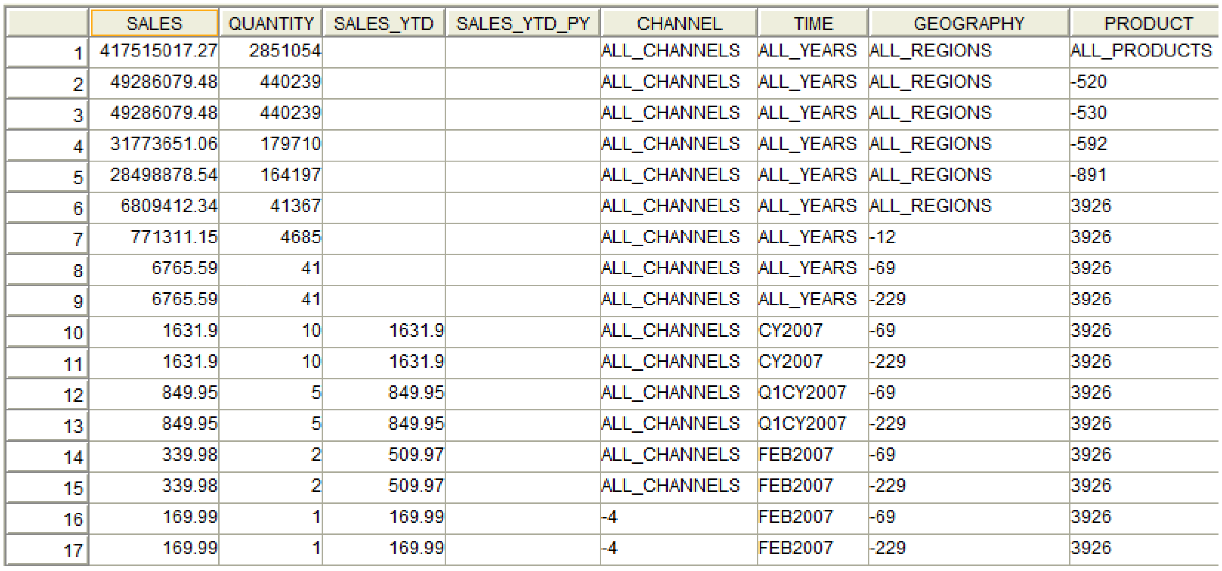


	DIM_KEY	LEVEL_NAME	MEMBER_TYPE	DIM_ORDER
1	18	CHANNEL	L	3
2	19	CHANNEL	L	4
3	20	CHANNEL	L	5
4	21	CHANNEL	L	6
5	22	CHANNEL	L	7
6	23	CHANNEL	L	8
7	24	CHANNEL	L	9
8	25	CHANNEL	L	10
9	26	CHANNEL	L	11
10	27	CHANNEL	L	12
11	28	CHANNEL	L	13
12	29	CHANNEL	L	14
13	30	CHANNEL	L	15
14	-4	CLASS	L	2
15	-3	CLASS	L	1
16	ALL_CHANNELS	ALL_CHANNELS	A	16



PARENT	DEPTH	ALL_CHANNELS	CLASS	CHANNEL
-4	2	ALL_CHANNELS	-4	18
-3	2	ALL_CHANNELS	-3	19
-3	2	ALL_CHANNELS	-3	20
-3	2	ALL_CHANNELS	-3	21
-4	2	ALL_CHANNELS	-4	22
-3	2	ALL_CHANNELS	-3	23
-3	2	ALL_CHANNELS	-3	24
-3	2	ALL_CHANNELS	-3	25
-3	2	ALL_CHANNELS	-3	26
-3	2	ALL_CHANNELS	-3	27
-3	2	ALL_CHANNELS	-3	28
-4	2	ALL_CHANNELS	-4	29
-3	2	ALL_CHANNELS	-3	30
ALL_CHANNE...	1	ALL_CHANNELS	-4	
ALL_CHANNE...	1	ALL_CHANNELS	-3	
	0	ALL_CHANNELS		

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:



	SALES	QUANTITY	SALES_YTD	SALES_YTD_PY	CHANNEL	TIME	GEOGRAPHY	PRODUCT
1	417515017.27	2851054			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	ALL_PRODUCTS
2	49286079.48	440239			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	-520
3	49286079.48	440239			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	-530
4	31773651.06	179710			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	-592
5	28498878.54	164197			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	-891
6	6809412.34	41367			ALL_CHANNELS	ALL_YEARS	ALL_REGIONS	3926
7	771311.15	4685			ALL_CHANNELS	ALL_YEARS	-12	3926
8	6765.59	41			ALL_CHANNELS	ALL_YEARS	-69	3926
9	6765.59	41			ALL_CHANNELS	ALL_YEARS	-229	3926
10	1631.9	10	1631.9		ALL_CHANNELS	CY2007	-69	3926
11	1631.9	10	1631.9		ALL_CHANNELS	CY2007	-229	3926
12	849.95	5	849.95		ALL_CHANNELS	Q1CY2007	-69	3926
13	849.95	5	849.95		ALL_CHANNELS	Q1CY2007	-229	3926
14	339.98	2	509.97		ALL_CHANNELS	FEB2007	-69	3926
15	339.98	2	509.97		ALL_CHANNELS	FEB2007	-229	3926
16	169.99	1	169.99		-4	FEB2007	-69	3926
17	169.99	1	169.99		-4	FEB2007	-229	3926

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.

3. 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).

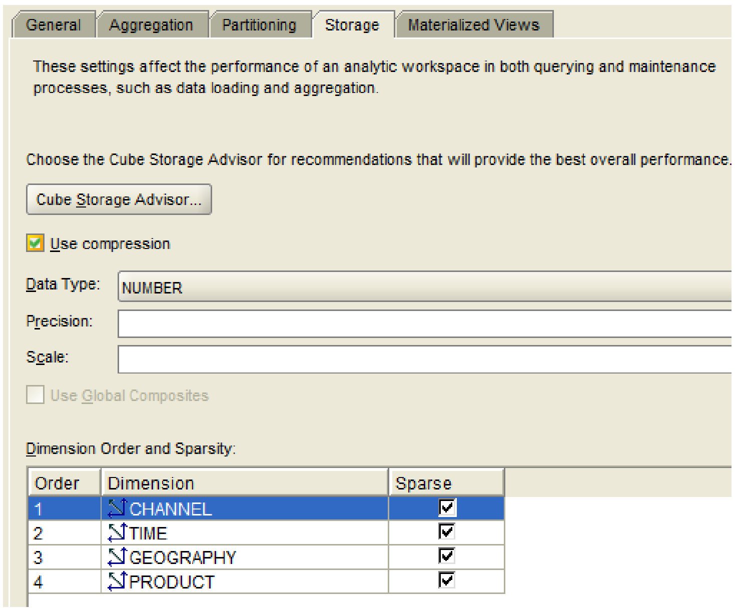
```
/* Check the count of records in the fact table. */
SELECT count(*) FROM SALES_FACT;

/* 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:



These settings affect the performance of an analytic workspace in both querying and maintenance processes, such as data loading and aggregation.

Choose the Cube Storage Advisor for recommendations that will provide the best overall performance.

[✓] Use compression

Data Type: NUMBER

Precision:

Scale:

[ ] Use Global Composites

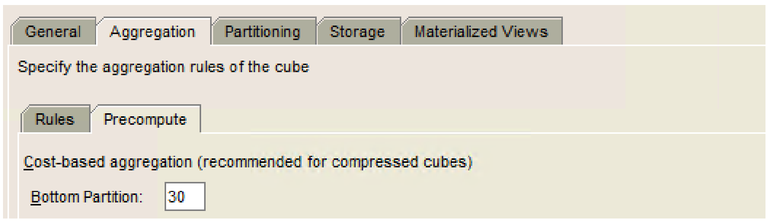
Dimension Order and Sparsity:

Order	Dimension	Sparse
1	CHANNEL	✓
2	TIME	✓
3	GEOGRAPHY	✓
4	PRODUCT	✓

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.

2) 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)


General | Aggregation | Partitioning | Storage | Materialized Views

Specify the aggregation rules of the cube

Rules | Precompute

Cost-based aggregation (recommended for compressed cubes)

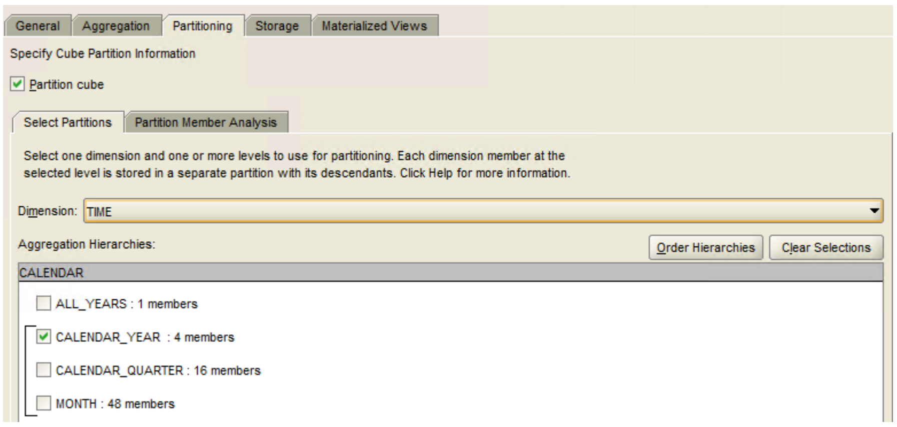
Bottom Partition: 30

3) 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.


General | Aggregation | Partitioning | Storage | Materialized Views

Specify Cube Partition Information

☑ Partition cube

Select Partitions | Partition Member Analysis

Select one dimension and one or more levels to use for partitioning. Each dimension member at the selected level is stored in a separate partition with its descendants. Click Help for more information.

Dimension: TIME

Aggregation Hierarchies:

Order Hierarchies | Clear Selections

CALENDAR

☐ ALL_YEARS : 1 members

☑ CALENDAR_YEAR : 4 members

☐ CALENDAR_QUARTER : 16 members

☐ MONTH : 48 members

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:

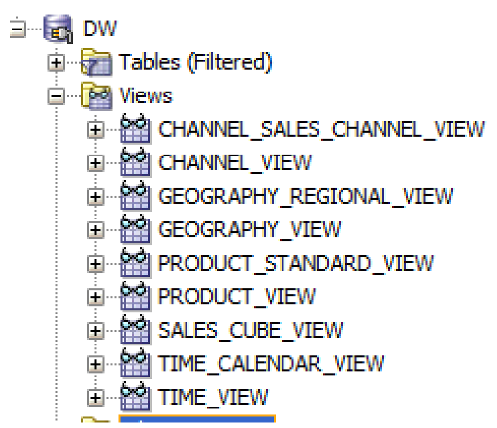
- Fact tables in a star schema store detailed data, while the cube views reveal many summary levels.
- In addition to the facts in fact tables, the cube provides additional measures and calculations, which are calculated and materialized as columns in the cube view.

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.

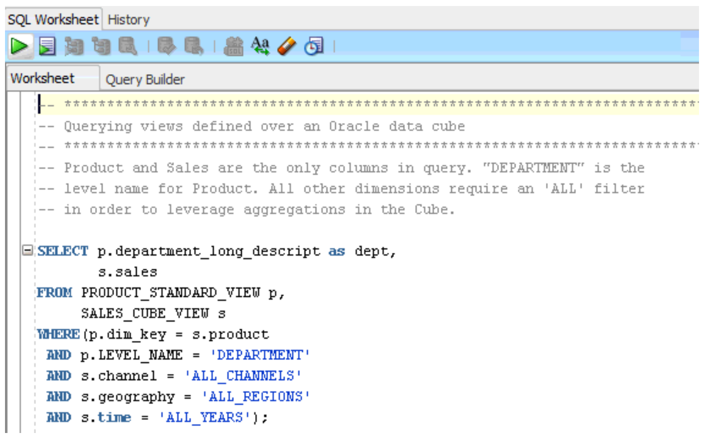
Since the cube data is made directly accessible to SQL by a set of relational views, in this part, we will write some SQL queries to query the cube views.

1. Simple Cube Queries

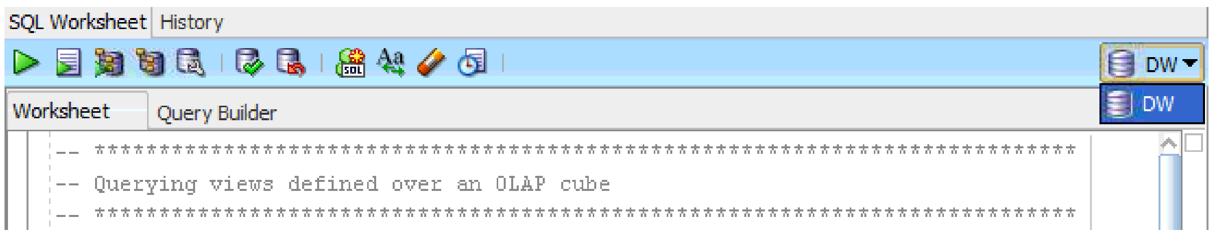
Get back to the **SQL Developer** and connect to the olaptrain schema. Check the views that were created in AWM.



```
DW
  Tables (Filtered)
  Views
    CHANNEL_SALES_CHANNEL_VIEW
    CHANNEL_VIEW
    GEOGRAPHY_REGIONAL_VIEW
    GEOGRAPHY_VIEW
    PRODUCT_STANDARD_VIEW
    PRODUCT_VIEW
    SALES_CUBE_VIEW
    TIME_CALENDAR_VIEW
    TIME_VIEW
```

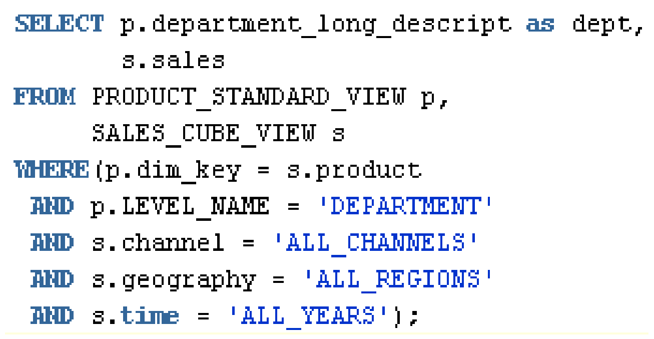


```
-- ****************************************************************************
-- Querying views defined over an Oracle data cube
-- ****************************************************************************
-- Product and Sales are the only columns in query. "DEPARTMENT" is the
-- level name for Product. All other dimensions require an 'ALL' filter
-- in order to leverage aggregations in the Cube.

SELECT p.department_long_descript as dept,
       s.sales
FROM PRODUCT_STANDARD_VIEW p,
     SALES_CUBE_VIEW s
WHERE(p.dim_key = s.product
 AND p.LEVEL_NAME = 'DEPARTMENT'
 AND s.channel = 'ALL_CHANNELS'
 AND s.geography = 'ALL_REGIONS'
 AND s.time = 'ALL_YEARS');
```



```
-- ****************************************************************************
-- Querying views defined over an OLAP cube
-- ****************************************************************************
```

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



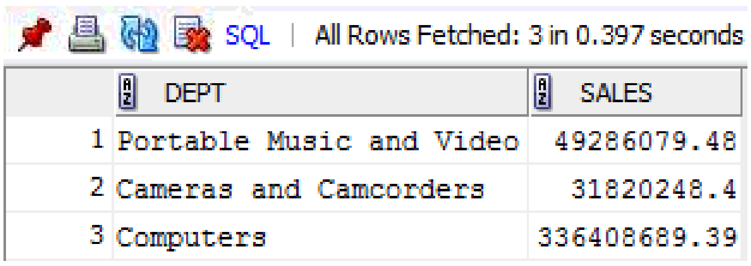
```
SELECT p.department_long_descript as dept,
       s.sales
FROM PRODUCT_STANDARD_VIEW p,
     SALES_CUBE_VIEW s
WHERE(p.dim_key = s.product
 AND p.LEVEL_NAME = 'DEPARTMENT'
 AND s.channel = 'ALL_CHANNELS'
 AND s.geography = 'ALL_REGIONS'
 AND s.time = 'ALL_YEARS');
```

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:



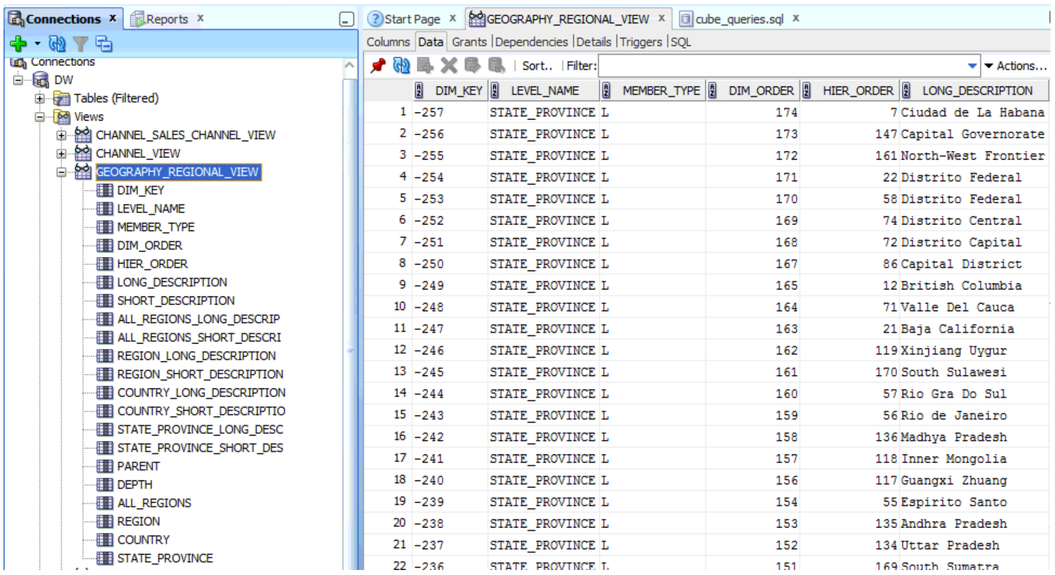
	DEPT	SALES
1	Portable Music and Video	49286079.48
2	Cameras and Camcorders	31820248.4
3	Computers	336408689.39

2. 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.



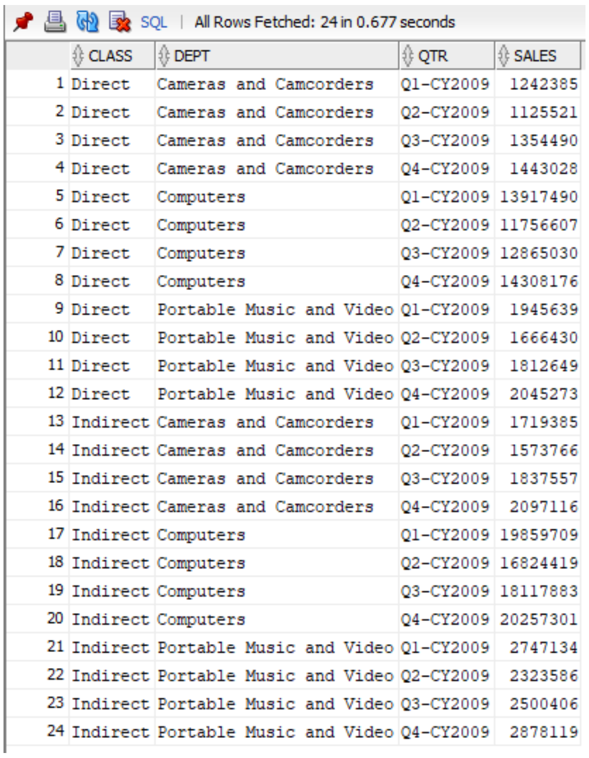
```
Connections
  DW
    Tables (Filtered)
    Views
      CHANNEL_SALES_CHANNEL_VIEW
      CHANNEL_VIEW
      GEOGRAPHY_REGIONAL_VIEW
        DIM_KEY
        LEVEL_NAME
        MEMBER_TYPE
        DIM_ORDER
        HIER_ORDER
        LONG_DESCRIPTION
        SHORT_DESCRIPTION
        ALL_REGIONS_LONG_DESCRIP
        ALL_REGIONS_SHORT_DESCRI
        REGION_LONG_DESCRIPTION
        REGION_SHORT_DESCRIPTION
        COUNTRY_LONG_DESCRIPTION
        COUNTRY_SHORT_DESCRIPTIO
        STATE_PROVINCE_LONG_DESC
        STATE_PROVINCE_SHORT_DES
        PARENT
        DEPTH
        ALL_REGIONS
        REGION
        COUNTRY
        STATE_PROVINCE
```

	DIM_KEY	LEVEL_NAME	MEMBER_TYPE	DIM_ORDER	HIER_ORDER	LONG_DESCRIPTION
1	-257	STATE_PROVINCE	L	174	7	Ciudad de La Habana
2	-256	STATE_PROVINCE	L	173	147	Capital Governorate
3	-255	STATE_PROVINCE	L	172	161	North-West Frontier
4	-254	STATE_PROVINCE	L	171	22	Distrito Federal
5	-253	STATE_PROVINCE	L	170	58	Distrito Federal
6	-252	STATE_PROVINCE	L	169	74	Distrito Central
7	-251	STATE_PROVINCE	L	168	72	Distrito Capital
8	-250	STATE_PROVINCE	L	167	86	Capital District
9	-249	STATE_PROVINCE	L	165	12	British Columbia
10	-248	STATE_PROVINCE	L	164	71	Valle Del Cauca
11	-247	STATE_PROVINCE	L	163	21	Baja California
12	-246	STATE_PROVINCE	L	162	119	Xinjiang Uygur
13	-245	STATE_PROVINCE	L	161	170	South Sulawesi
14	-244	STATE_PROVINCE	L	160	57	Rio Gra Do Sul
15	-243	STATE_PROVINCE	L	159	56	Rio de Janeiro
16	-242	STATE_PROVINCE	L	158	136	Madhya Pradesh
17	-241	STATE_PROVINCE	L	157	118	Inner Mongolia
18	-240	STATE_PROVINCE	L	156	117	Guangxi Zhuang
19	-239	STATE_PROVINCE	L	154	55	Espirito Santo
20	-238	STATE_PROVINCE	L	153	135	Andhra Pradesh
21	-237	STATE_PROVINCE	L	152	134	Uttar Pradesh
22	-236	STATE_PROVINCE	L	151	169	South Sumatra

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* dimensions and *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):


SQL | All Rows Fetched: 24 in 0.677 seconds

	CLASS	DEPT	QTR	SALES
1	Direct	Cameras and Camcorders	Q1-CY2009	1242385
2	Direct	Cameras and Camcorders	Q2-CY2009	1125521
3	Direct	Cameras and Camcorders	Q3-CY2009	1354490
4	Direct	Cameras and Camcorders	Q4-CY2009	1443028
5	Direct	Computers	Q1-CY2009	13917490
6	Direct	Computers	Q2-CY2009	11756607
7	Direct	Computers	Q3-CY2009	12865030
8	Direct	Computers	Q4-CY2009	14308176
9	Direct	Portable Music and Video	Q1-CY2009	1945639
10	Direct	Portable Music and Video	Q2-CY2009	1666430
11	Direct	Portable Music and Video	Q3-CY2009	1812649
12	Direct	Portable Music and Video	Q4-CY2009	2045273
13	Indirect	Cameras and Camcorders	Q1-CY2009	1719385
14	Indirect	Cameras and Camcorders	Q2-CY2009	1573766
15	Indirect	Cameras and Camcorders	Q3-CY2009	1837557
16	Indirect	Cameras and Camcorders	Q4-CY2009	2097116
17	Indirect	Computers	Q1-CY2009	19859709
18	Indirect	Computers	Q2-CY2009	16824419
19	Indirect	Computers	Q3-CY2009	18117883
20	Indirect	Computers	Q4-CY2009	20257301
21	Indirect	Portable Music and Video	Q1-CY2009	2747134
22	Indirect	Portable Music and Video	Q2-CY2009	2323586
23	Indirect	Portable Music and Video	Q3-CY2009	2500406
24	Indirect	Portable Music and Video	Q4-CY2009	2878119

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