Data Science And Database Technology

Laboratory Practise 3

Materialized view and trigger

The purpose of this tutorial is to define materialized views that are useful for quickly responding to frequent queries from the data warehouse.
The views created must then be kept updated appropriately, managing any changes made on the initial tables of the data warehouse.

1. Connection to the database

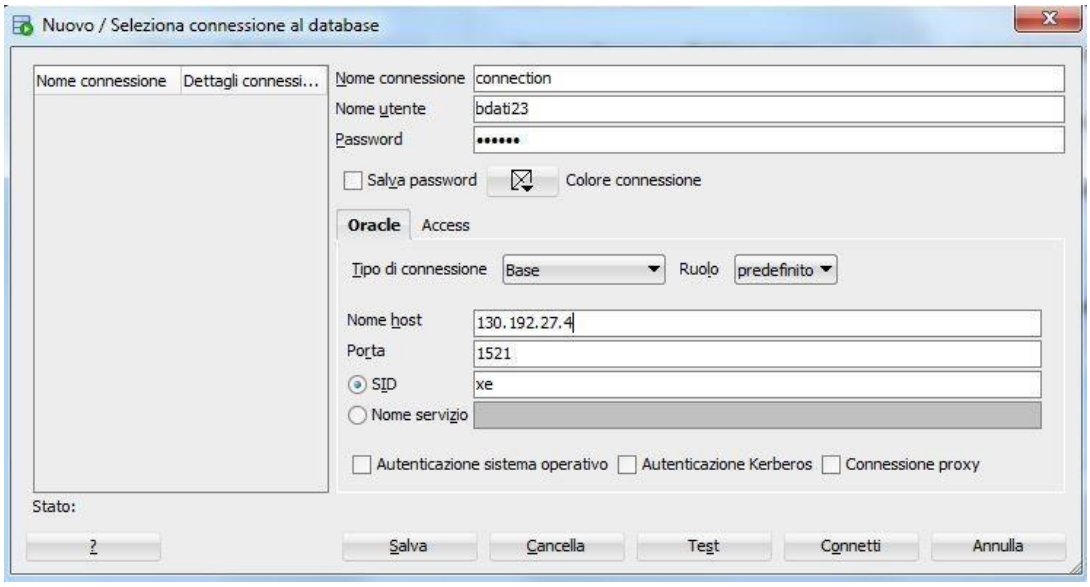
Open the Oracle SQL Developer program and create a connection to the Oracle database as you did in the first lab practise.

Login

a. To logon you have to insert the following parameters:

i. Nome utente (username): bdati[choose a number between 1-100]
ii. Password: orac[choose a number between 1-100]
iii. Nome host (host name): 130.192.27.4
iv. Port: 1521
v. SID: xe

For example, if you are working on pc number 23, the corresponding username is bdati23 and the password is orac23.



Populate the database with the data warehouse tables used in the first practise (download the zip on the course website). Make sure your database does not already contain the tables (FACTS, LOCATION, PHONERATE, TIMEDIM). If they already exist, they must be removed and then import the downloaded files in csv format.

2. Creation and updating of the materialized view with the use of CREATE MATERIALIZED VIEW in ORACLE

2.1. Creation and update of materialized view

Exercise 1.1 Starting from the data warehouse described in the first laboratory practuse (and whose logical scheme is shown in the Table 1), define two materialized views useful for reducing the response times of at least three of the 6 queries listed below.

Tables	Description
DWABD.TIMEDIM (ID_time INT NOT NULL, DateDay DATE NOT NULL, DayOfWeek CHAR(15) NOT NULL, DateMonth CHAR(15) NOT NULL, DateYear INT NOT NULL, PRIMARY KEY(ID_time));	Time dimension 10 rows
DWABD.PHONERATE (ID_phoneRate INTEGER NOT NULL, phoneRateType VARCHAR(20) NOT NULL, PRIMARY KEY(ID_phoneRate));	Phone rate dimension 7 rows
DWABD.LOCATION (ID_location INTEGER NOT NULL, City VARCHAR(20) NOT NULL, Province CHAR(20) NOT NULL, Region CHAR(20) NOT NULL, PRIMARY KEY(ID_location));	Place dimension 1500 rows
DWABD.FACTS (ID_time INTEGER NOT NULL, ID_phoneRate INTEGER NOT NULL, ID_location_Caller INTEGER NOT NULL, ID_location_Receiver INTEGER NOT NULL, Price FLOAT NOT NULL, NumberOfCalls INTEGER NOT NULL, PRIMARY KEY(ID_time,ID_phoneRate,ID_location_Caller,ID_location_Receiver), FOREIGN KEY(ID_time) REFERENCES timeDim(ID_time), FOREIGN KEY(ID_phoneRate) REFERENCES phoneRate(ID_phoneRate), FOREIGN KEY(ID_location_Caller) REFERENCES location(ID_location), FOREIGN KEY(ID_location_Receiver) REFERENCES location(ID_location));	Fact table 7809 rows

Table 1 – Data warehouse tables

SQL QUERIES:

1. Select the total income for each type of phone rate and for each month of the year 2003. Also select the total income, the total income for each type of phone rate regardless of the month, and the total income for each month regardless of the type of phone rate.
2. Select the monthly number of calls and the monthly income. Associate the RANK() to each month according to its income (1 for the month with the highest income, 2 for the second, etc., the last month is the one with the least income).
3. For each month in 2003, select the total number of calls. Associate the RANK() to each month according to its total number of calls (1 for the month with the highest number of calls, 2 for the second, etc., the last month is the one with the least number of calls).
4. Separately by phone rate, select the total income of January 2003.
5. Select the monthly income and the cumulative monthly income from the beginning of the year.
6. Consider the year 2003. Separately for phone rate and month, analyze (i) the total income, (ii) the percentage of income with respect to the total revenue considering all the phone rates, (iii) the percentage of income with respect to the total revenue considering all the months.

Exercise 1.2 After creating the materialized views, make sure that these are suitably updated when any changes occur on the data. Which tables should be monitored to update the views created accordingly?

Follow the steps below to create the logs and their view:

STEP A: Try to modify the contents of the FACTS table as follows:

```
insert into FACTS(Id_time, ID_phoneRate, ID_location_Caller, ID_location_Receiver, Price,
NumberOfCalls)     values(8,1,558,752,40000,150)

insert into FACTS(Id_time, ID_phoneRate, ID_location_Caller, ID_location_Receiver, Price,
NumberOfCalls)    values(1,6,558,752,100,100)
```

STEP B: Now check the new content of the materialized views, updating them with the following command:

```
BEGIN
DBMS_SNAPSHOT.REFRESH ('View_Name');
END;
```

How have the two materialized views changed?

Exercise 1.3 (Optional) Try running the queries with and without materialized views and verify that the output obtained is the same in both cases.

Exercise 1.4 (Optional for those who carry out the exercise on their PC) Try to update the content of the materialized views, creating the materialized view log for all the necessary tables, in order to monitor the changes of each table. For which tables do you need to create the appropriate log structures?

When a new record is inserted in the FACTS table, the materialized views linked to this table must be updated using the FAST REFRESH option. In Oracle, to use this option, **materialized view logs must be created** for all the necessary tables **before creating the view**, in order to monitor the changes of each table.

STEP A:

For each table where you deem it necessary, create the view log as follows:

```
CREATE MATERIALIZED VIEW LOG ON TABLE1
  WITH ROWID, SEQUENCE(Attribute_1, Attribute_2, Attribute_3)
  INCLUDING NEW VALUES;
```

STEP B: Create the materialized view, making sure that the update method option is set to "FAST" and that the update type is "ON COMMIT".

STEP C: Change the fact table as indicated in exercise 1.2 and verify that the change is correctly propagated on the materialized views.
Make sure that the option relating to the type of update is still set to "FAST" (right click on the view)

N.B: Materialized view logs cannot be created with the version of Oracle present on the lab computers.

3. Update and management of views via Trigger

Assuming that the CREATE MATERIALIZED VIEW command is not available, now create the materialized views defined in the previous exercise by following the steps listed:

STEP 3.1 Create the structure of the materialized view with the following statement:

```
CREATE TABLE VM1 (…)
```

STEP 3.2 Populate the VM1 table with the necessary records using the following statement

```
INSERT INTO VM1 (…)
( SELECT …
  … )
```

Exercise: Now write the Triggers necessary to propagate the changes (insertion of a new record) made in the FACTS table to the materialized views created VM1 and VM2.

Verify that the triggers are working correctly by performing the following and verifying that VM1 and VM2 are updated accordingly:

```
insert into FACTS(Id_time, ID_phoneRate, ID_location_Caller, ID_location_Receiver, Price,
NumberOfCalls)  values(8,2,558,752,40000,150)

insert into FACTS(Id_time, ID_phoneRate, ID_location_Caller, ID_location_Receiver, Price,
NumberOfCalls) values(1,7,558,752,100,100)
```

On which of the two tables has an existing record been updated? On which one has a new record been inserted?

Useful statements for trigger management:

- Drop a trigger:
  - drop trigger trigger_name;
- Update or substitute an existing trigger:
  - CREATE OR REPLACE TRIGGER trigger_name;
- Visualized the created trigger:
  - select trigger_name, triggering_event, table_name, status, description, action_type, trigger_body from user_triggers;
- Visualize trigger errors:
  - select * from USER_ERRORS;