Devecor BI suite

25Dec2019
Advance Db PL/SQL

MOPIDEVI Murali Krishna
SINGH Arjun

Table of Contents

Introductions

The purpose of this project is to implement the learnings from PL/SQL and apply it to demo case of DEVECOR. Our objective is to build operational database do the ETL and build warehouse with DataMart's. But for this case we replicate the framework of data warehouse with 2 databases and pushing the data after cleaning and then building the small tables needed by the respective department

What is Data Warehouse??

A “data warehouse” is a repository of historical data that is organized by subject to support decision makers in an organization. Data warehouses are systems used to store data from one or more disparate sources in a centralized place where it can be accessed for reporting and data analytics. The data in the data warehouse may be current or historical, and may be in its original raw data form or processed/ summarized.

The data in a data warehouse is imported from source systems (such as ERP, CRM or Finance platforms) and gathered in the warehouse where it can be used across the enterprise for creating analytical reports and to support business decision-making. The general process used to aggregate and transform data for warehousing is referred to as “extract, transform and load,” or ETL for short. What this means is a company takes a copy of data from source systems, leaving the original data intact and in place – avoiding disruption to transactional processes refined and processed to remove data that may be occurring.

Once data is loaded into the data warehouse, it is further quality issues, integrate interdependent data sources and organize it for ease of consumption. Data warehouses also often contain pre-processed summaries of data and snapshots of data from different points in time that are used to assist in analysis. Where transactional systems are most concerned with maintaining the current state of data (and do so by overwriting values when data is updated), warehouses maintain the history of how a company’s data changes and evolves. This is particularly important when conducting trend analysis and other business analytics intended to answer questions about “why” something happened within a company.

*Why do companies need **Data Warehouse?***

1. The need for information insights that span multiple source systems
2. Provide a long-term archive for transactional data, so source systems can be purged to maintain high performance
3. To provide a place where reporting and analytics can occur without creating an additional load on operational systems

Project Requirements

1. ***Decisional DB*** *are dedicated to the business of the company*
2. ***Operational DB*** *are dedicated to the day to day operations of the company*
3. ***Accessibility*** *based on profile and privileges*
4. ***A breakdown of sales and turnover*** *over different granularity of time (week, month, year etc…)*

PROJECT TASKS

1. ***Operational database for company***
   - *The script for creating tables in the operational database*
   - *The script of my procedure that will generate the dataset.*
   - *And finally you give the privileges to the session 2 to read the contents of the created tables*

2. ***Decisional database for the management of the company***
   - *The script for creating the tables of the decision database*
   - *The script of the procedure that will select to give them from the operational database to put them in the decision-making database (ETL).*

3. ***Expression of needs***

   The sales manager wants:
   - *To study the turnover and sales volume*
   - *By product and Family.*
   - *Per week, month and year.*
   - *By department and region.*

4. ***A procedure to generate***
   - *receives two dates and supplies the fact table from operational for all orders made between two dates*

SCRIPTS

User Scripts

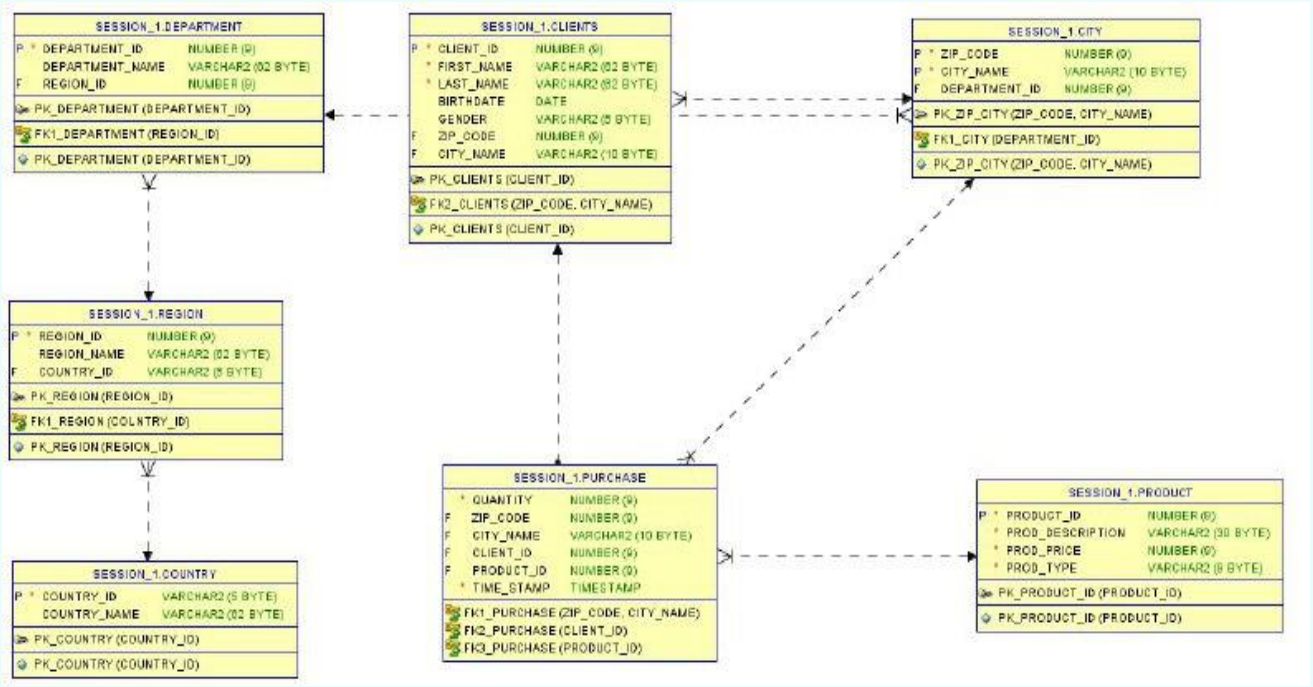
```
================================================================
-- CONNECT AS A SYSTEM ADMIN AND CREATE 2 USERS
================================================================

 CONNECT SYSTEM/SYSTEM;
================================================================
-- CREATING SESSION_1
================================================================
CREATE  USER session_1 IDENTIFIED BY session_1;
GRANT  CONNECT, RESOURCE ,dba TO session_1;

CONNECT session_1/session_1;

--GRANT SELECT ANY TABLE TO Session_2;
GRANT SELECT ON COUNTRY TO session_2;
GRANT SELECT ON CITY TO session_2;
GRANT SELECT ON CLIENTS TO session_2;
GRANT SELECT ON DEPARTMENT TO session_2;
GRANT SELECT ON PRODUCT TO session_2;
GRANT SELECT ON PURCHASE TO session_2;
GRANT SELECT ON REGION TO session_2;

DISCONNECT;
================================================================
-- CREATING SESSION_2
================================================================
CONNECT SYSTEM/SYSTEM;

CREATE  USER session_2 IDENTIFIED BY session_2;
GRANT  CONNECT, RESOURCE TO session_2;

CONNECT session_2/session_2;

DISCONNECT;

================================================================
-- PLEASE CHECK
-- OPERATIONALDB.SQL
-- DECISIONALDB.SQL
-- FOR INSERTING DATA INTO DATABASES.
```

Operational database

```
================================================================
-- Table : COUNTRY
================================================================
CREATE TABLE COUNTRY
(
  COUNTRY_ID  VARCHAR(5)  NOT NULL,
  COUNTRY_NAME  VARCHAR2(62) ,
  CONSTRAINT pk_country PRIMARY KEY (COUNTRY_ID)
);
================================================================
-- Table : REGION
================================================================
CREATE TABLE REGION
(
  REGION_ID  NUMBER(9)  NOT NULL,
  REGION_NAME  VARCHAR2(62) ,
  COUNTRY_ID VARCHAR(5),
  CONSTRAINT pk_region PRIMARY KEY (REGION_ID)
);
================================================================
-- Table : DEPARTMENT
================================================================
CREATE TABLE DEPARTMENT
(
  DEPARTMENT_ID  NUMBER(9)  NOT NULL,
  DEPARTMENT_NAME VARCHAR2(62) ,
  REGION_ID  NUMBER(9),
  CONSTRAINT pk_department PRIMARY KEY (DEPARTMENT_ID)
);
================================================================
-- Table : CITY
================================================================
CREATE TABLE CITY
(
ZIP_CODE NUMBER(9) NOT NULL,
CITY_NAME  VARCHAR2(10) ,
DEPARTMENT_ID  NUMBER(9),
CONSTRAINT pk_zip_city PRIMARY KEY (ZIP_CODE, CITY_NAME)
);
================================================================
-- Table : CLIENT
================================================================
CREATE TABLE CLIENTS
(
  CLIENT_ID  NUMBER(9)  NOT NULL,
  FIRST_NAME  VARCHAR(62)  NOT NULL,
  LAST_NAME  VARCHAR(62)  NOT NULL,
  BIRTHDATE  DATE,
  GENDER  VARCHAR2(5) ,
  ZIP_CODE        NUMBER(9),
  CITY_NAME  VARCHAR2(10) ,
  CONSTRAINT  pk_clients PRIMARY KEY (CLIENT_ID)
);
```

```
================================================================
-- Table : PRODUCT
================================================================
CREATE TABLE PRODUCT
(
  PRODUCT_ID  NUMBER(9)  NOT NULL,
  PROD_DESCRIPTION   VARCHAR2(30)   NOT NULL,
  PROD_PRICE   NUMBER(9)  NOT NULL,
  PROD_TYPE  VARCHAR2(9) NOT NULL,
  CONSTRAINT pk_product_id PRIMARY KEY (PRODUCT_ID)
);
================================================================
-- Table : PURCHASE
================================================================
CREATE TABLE PURCHASE
(
  QUANTITY  NUMBER(9)   NOT NULL,
  ZIP_CODE          NUMBER(9),
  CITY_NAME  VARCHAR2(10),
  CLIENT_ID  NUMBER(9),
  PRODUCT_ID  NUMBER(9),
  TIME_STAMP  TIMESTAMP  NOT NULL);
================================================================
-- FOREIGN KEYS : ALL TABLES
-- COUNTRY, REGION, DEPARTMENT, CITY, CLIENT, DELIVERY, PRODUCT
================================================================
ALTER TABLE REGION ADD CONSTRAINT fk1_region FOREIGN KEY (COUNTRY_ID)
    REFERENCES COUNTRY (COUNTRY_ID);
----------------------------------------------------------------------------
ALTER TABLE DEPARTMENT ADD CONSTRAINT fk1_department  FOREIGN KEY (REGION_ID)
    REFERENCES REGION (REGION_ID);
----------------------------------------------------------------------------
ALTER TABLE  CITY  ADD CONSTRAINT fk1_city  FOREIGN KEY (DEPARTMENT_ID)
    REFERENCES DEPARTMENT (DEPARTMENT_ID);
----------------------------------------------------------------------------
ALTER TABLE  CLIENTS  ADD CONSTRAINT fk2_clients FOREIGN KEY (ZIP_CODE, CITY_NAME)
    REFERENCES CITY (ZIP_CODE, CITY_NAME);
----------------------------------------------------------------------------
ALTER TABLE  PURCHASE  ADD CONSTRAINT fk1_purchase FOREIGN KEY (ZIP_CODE, CITY_NAME)
    REFERENCES CITY (ZIP_CODE, CITY_NAME);
----------------------------------------------------------------------------
ALTER TABLE PURCHASE  ADD CONSTRAINT fk2_purchase FOREIGN KEY (CLIENT_ID)
    REFERENCES CLIENTS(CLIENT_ID);
----------------------------------------------------------------------------
ALTER TABLE  PURCHASE ADD CONSTRAINT fk3_purchase  FOREIGN KEY (PRODUCT_ID)
    REFERENCES PRODUCT(PRODUCT_ID);
================================================================
-- CREATING SEQUENCE
================================================================
CREATE SEQUENCE seq_country;
CREATE SEQUENCE seq_region;
CREATE SEQUENCE seq_department;
CREATE SEQUENCE seq_city;
CREATE SEQUENCE seq_clients;
CREATE SEQUENCE seq_product;
CREATE SEQUENCE seq_purchase;
================================================================
-- TURNING SERVER ON
  SET SERVEROUTPUT ON
```

Data Model



Procedures to populate the database , use of more and more realistic random data to have that readability in my tables, as such use of random values like 'asdkjkjdhjdsflkjs' we have used randomness like Dept_ 1, city_1 for countries we have used actual countries like india , usa , france(ref sql file: 02_operational_DB.sql

Sample procedure for operational db

```
CREATE OR REPLACE PROCEDURE data_purchase  AS
   M_ZIP_CODE     NUMBER(9);
   M_CITY_NAME  VARCHAR2(10);
   M_CLIENT_ID  NUMBER(9);
   M_QUANTITY  NUMBER(9);
   M_TIME_STAMP  TIMESTAMP;
   M_PRODUCT_ID  NUMBER(9);
   num_1 INTEGER;
BEGIN

 FOR i IN 1..5000
 LOOP
    ------------ CREATING RANDOM QUANTITY  ------------
              num_1 := round(dbms_random.value(1,100));

              M_QUANTITY  := num_1;
  SELECT ZIP_CODE, CITY_NAME INTO M_ZIP_CODE,M_CITY_NAME FROM ( SELECT
city.ZIP_CODE, city.CITY_NAME FROM city WHERE city.CITY_NAME LIKE '%'
  ORDER BY dbms_random.value ) WHERE ROWNUM < 2;
```

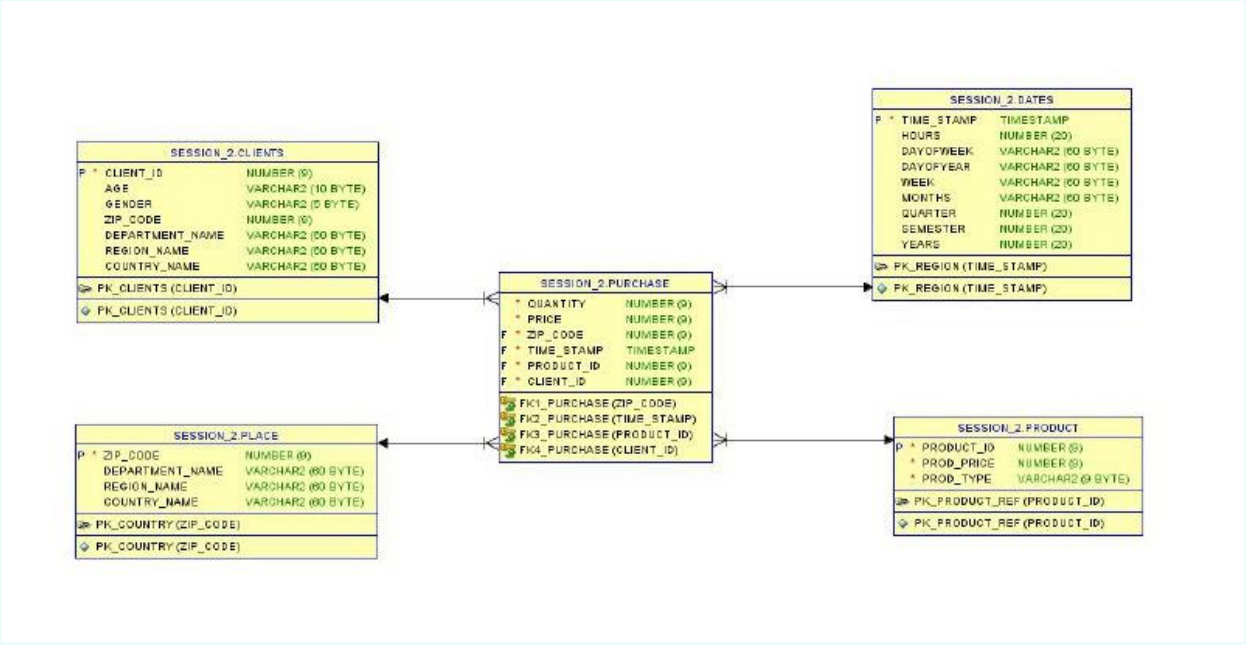
```
  SELECT CLIENT_ID INTO M_CLIENT_ID FROM ( SELECT CLIENT_ID FROM CLIENTS ORDER BY
dbms_random.value ) WHERE rownum = 1;
  SELECT PRODUCT_ID INTO M_PRODUCT_ID FROM ( SELECT PRODUCT_ID FROM PRODUCT
ORDER BY dbms_random.value ) WHERE rownum = 1;
  SELECT TO_DATE('01-01-2009','mm-dd-yyyy') + dbms_random.VALUE(0,86400*(to_date('01-JAN-
2009', 'dd-mm-yyyy') -
    to_date('10-DEC-2019', 'dd-mm-yyyy'))+1)/86400 INTO M_TIME_STAMP FROM dual;
  INSERT INTO purchase VALUES(M_QUANTITY,M_ZIP_CODE,M_CITY_NAME, M_CLIENT_ID,
M_PRODUCT_ID,M_TIME_STAMP);
 end loop
END data_purchase;
/
EXEC data_purchase;
```

Decisional database

```
-- =============================================================
--   TABLE : PLACE
-- =============================================================
CREATE TABLE PLACE
(
   ZIP_CODE NUMBER(9) NOT NULL,
    DEPARTMENT_NAME VARCHAR2(60),
    REGION_NAME VARCHAR2(60),
   COUNTRY_NAME VARCHAR2(60),
   CONSTRAINTS pk_country PRIMARY KEY (ZIP_CODE)
);
-- =============================================================
--   TABLE : PRODUCT
-- =============================================================
CREATE TABLE PRODUCT
(
   PRODUCT_ID NUMBER(9) NOT NULL,
   PROD_PRICE NUMBER(9) NOT NULL,
   PROD_TYPE VARCHAR2(9) NOT NULL,
   CONSTRAINTS pk_product_ref PRIMARY KEY (PRODUCT_ID)
);
-- =============================================================
--   TABLE : DATES
-- =============================================================
CREATE TABLE DATES
(
   TIME_STAMP TIMESTAMP NOT NULL,
   HOURS NUMBER(20),
    DAYOFWEEK VARCHAR2(60),
    DAYOFYEAR VARCHAR2(60),
    WEEK VARCHAR2(60),
    MONTHS VARCHAR2(60),
    QUARTER NUMBER(20),
    SEMESTER NUMBER(20),
    YEARS NUMBER(20),
    CONSTRAINTS pk_region PRIMARY KEY (TIME_STAMP)
);
-- =============================================================
--   TABLE : CLIENTS
-- =============================================================
CREATE TABLE CLIENTS
(
   CLIENT_ID NUMBER(9) NOT NULL,
```

```
   AGE VARCHAR2(10),
   GENDER VARCHAR2(5),
    ZIP_CODE NUMBER(9),
    DEPARTMENT_NAME VARCHAR2(60),
    REGION_NAME VARCHAR2(60),
    COUNTRY_NAME VARCHAR2(60),
   CONSTRAINTS pk_clients PRIMARY KEY (CLIENT_ID)
);
-- ============================================================
--   TABLE : PURCHASE
-- ============================================================
CREATE TABLE PURCHASE
(
   QUANTITY NUMBER(9) NOT NULL,
   PRICE NUMBER(9) NOT NULL,
    ZIP_CODE NUMBER(9) NOT NULL,
    TIME_STAMP TIMESTAMP NOT NULL,
    PRODUCT_ID NUMBER(9) NOT NULL,
    CLIENT_ID NUMBER(9)NOT NULL
);
-- ============================================================
--   FOREIGN KEYS : ALL TABLES
--   PLACE, PRODUCT, PURCHASE, DATES, CLIENT
-- ============================================================
ALTER TABLE PURCHASE ADD CONSTRAINTS fk1_purchase FOREIGN KEY (ZIP_CODE)
     REFERENCES PLACE(ZIP_CODE);

ALTER TABLE PURCHASE ADD CONSTRAINTS fk2_purchase FOREIGN KEY (TIME_STAMP)
     REFERENCES DATES(TIME_STAMP);

ALTER TABLE PURCHASE ADD CONSTRAINTS fk3_purchase FOREIGN KEY (PRODUCT_ID)
     REFERENCES PRODUCT(PRODUCT_ID);

ALTER TABLE PURCHASE ADD CONSTRAINTS fk4_purchase FOREIGN KEY (CLIENT_ID)
     REFERENCES CLIENTS(CLIENT_ID);
```

Data Model



Note: Now to populate the decisional db tables we have used trigger to instantiate the insertion

```
================================================================
-- Creating trigger in session_2
================================================================
CREATE OR REPLACE TRIGGER insert_data_product
AFTER INSERT ON SESSION_1.product
FOR EACH ROW
BEGIN
INSERT INTO SESSION_2.product (PRODUCT_ID,PROD_PRICE,PROD_TYPE) VALUES
(:NEW.PRODUCT_ID,:NEW.PROD_PRICE,:NEW.PROD_TYPE);
END;
/

================================================================
-- CREATING A PROCEDURE INSERTING DATA INTO ALL DECISIONAL TABLES
================================================================
CREATE OR REPLACE PROCEDURE data_decisional  AS
    BEGIN

================================================================
-- PRODUCT TABLE
================================================================
INSERT INTO PRODUCT (PRODUCT_ID,PROD_PRICE,PROD_TYPE)
SELECT PRODUCT_ID, PROD_PRICE,PROD_TYPE FROM session_1.PRODUCT ORDER BY
PRODUCT_ID ASC;

================================================================
-- PLACE TABLE
================================================================
    INSERT INTO PLACE (ZIP_CODE, DEPARTMENT_NAME, REGION_NAME, COUNTRY_NAME)
                        SELECT
                        city.ZIP_CODE AS ZIP_CODE
                        , department.DEPARTMENT_NAME AS DEPARTMENT_NAME
                        , region.REGION_NAME AS REGION_NAME
                        , country.COUNTRY_NAME AS COUNTRY_NAME
                        FROM
                                session_1.city
                                        INNER JOIN session_1.department
                                                USING(DEPARTMENT_ID)
                                        INNER JOIN session_1.region
                                                USING(REGION_ID)
                                        INNER JOIN session_1.country
                                                USING(COUNTRY_ID)
                                        ORDER BY
                                                ZIP_CODE ASC;

================================================================
-- CLIENTS TABLE
================================================================
INSERT INTO CLIENTS (CLIENT_ID, AGE, GENDER, ZIP_CODE, DEPARTMENT_NAME,
REGION_NAME, COUNTRY_NAME)
SELECT  clients.CLIENT_ID AS CLIENT_ID,
    round(months_between(TRUNC(to_date(clients.BIRTHDATE,'DD-MON-YY')),(sysdate))/12 ) as AGE,
                        clients.GENDER AS GENDER,
                        city.ZIP_CODE AS ZIP_CODE,
```

```
                department.DEPARTMENT_NAME AS DEPARTMENT_NAME,
                region.REGION_NAME AS REGION_NAME,
                country.COUNTRY_NAME AS COUNTRY_NAME
               FROM
                    session_1.clients
                        INNER JOIN session_1.city
                            ON clients.ZIP_CODE = city.ZIP_CODE
                        INNER JOIN session_1.department
                            USING(DEPARTMENT_ID)
                        INNER JOIN session_1.region
                            USING(REGION_ID)
                        INNER JOIN session_1.country
                            USING(COUNTRY_ID)
                        ORDER BY
                            CLIENT_ID ASC;

==========================================================================
-- DATE TABLE
==========================================================================
INSERT INTO DATES (TIME_STAMP, HOURS, DAYOFWEEK, DAYOFYEAR, WEEK,
MONTHS,QUARTER,SEMESTER,YEARS)
                            SELECT TIME_STAMP AS TIME_STAMP,
                            EXTRACT (HOUR FROM TIME_STAMP) AS HOURS,
                            TO_CHAR(TIME_STAMP,'DY') AS DAYOFWEEK,
                            TO_CHAR(TIME_STAMP, 'DDD') AS DAYOFYEAR,
                            TO_CHAR(TIME_STAMP, 'IW') AS WEEK,
                            TO_CHAR(TIME_STAMP,'MON')AS MONTHS,
                            TO_CHAR(TIME_STAMP, 'Q') AS QUARTER,
                            (CEIL(EXTRACT(MONTH FROM TIME_STAMP)/6)) AS
SEMESTER,
                            TO_CHAR(TIME_STAMP, 'IYYY') AS YEARS
                                FROM session_1.PURCHASE;

==========================================================================
-- PURCHASE TABLE - FACT TABLE
==========================================================================
            INSERT INTO PURCHASE (QUANTITY, PRICE, ZIP_CODE,
TIME_STAMP,PRODUCT_ID,CLIENT_ID)
            SELECT PURCHASE.QUANTITY AS QUANTITY,
                    PRODUCT.PROD_PRICE AS PRICE,
                    PURCHASE.ZIP_CODE AS ZIP_CODE,
                    PURCHASE.TIME_STAMP AS TIME_STAMP,
                    PURCHASE.PRODUCT_ID AS PRODUCT_ID,
                    PURCHASE.CLIENT_ID AS CLIENT_ID
             FROM session_1.PURCHASE
                    INNER JOIN session_1.PRODUCT
                            ON PURCHASE.PRODUCT_ID = PRODUCT.PRODUCT_ID
                                ORDER BY CLIENT_ID ASC;

END data_decisional;
/
EXEC data_decisional;
```

Note: So all in all one trigger to instantiate the insertion into decisional database and one procedure to insert into all tables which makes execution very streamlined

==========

Expression of needs

==========

```
--The sales manager wants:

-- ----------------------------To study the turnover and sales volume ------------------------
    -- By product and Family.
SELECT PRODUCT.PRODUCT_ID,
      PRODUCT.PROD_TYPE,
      (PURCHASE.QUANTITY*PURCHASE.PRICE) AS TURNOVER
      FROM PURCHASE
                  INNER JOIN PRODUCT
                        ON  PURCHASE.PRODUCT_ID = PRODUCT.PRODUCT_ID
                              ORDER BY PRODUCT.PROD_TYPE ASC;
-- Per week, month and year.
SELECT  PRODUCT_ID,
            TO_CHAR(TIME_STAMP, 'IW') AS WEEKS,
            TO_CHAR(TIME_STAMP, 'MON') AS MONTHS,
            TO_CHAR(TIME_STAMP, 'IYYY') AS YEARS,
            (QUANTITY*PRICE) AS TURNOVER
                  FROM PURCHASE
                        ORDER BY PRODUCT_ID ASC;

-- By department and region.
SELECT PLACE.DEPARTMENT_NAME AS DEPARTMENT,
      PLACE.REGION_NAME AS REGION,
      (PURCHASE.QUANTITY*PURCHASE.PRICE) AS TURNOVER
                  FROM PLACE
                        INNER JOIN PURCHASE
                              ON PLACE.ZIP_CODE = PURCHASE.ZIP_CODE
                                    ORDER BY PLACE.DEPARTMENT_NAME ASC;
```

==========

Facts Table

==========

```
-- We log on the system account (system , eistI0001)

-- CONNECTING TO SYSTEM
   CONN SYSTEM/SYSTEM;

-- CREATING TABLESPACE TO SESSION_1
CREATE TABLESPACE TS_session_1 datafile'H:\Masters\ADEO-2\1-1\Advance Database\PROJECT-
FINAL\FINAL\TS_session_1.dbf' size 3M;

-- CREATING TABLESPACE TO SESSION_2
CREATE TABLESPACE TS_session_2 datafile'H:\Masters\ADEO-2\1-1\Advance Database\PROJECT-
FINAL\FINAL\TS_session_2.dbf' size 3M;

-- ASSIGNING TABLESPACE TO SESSION_1
ALTER USER session_1  DEFAULT TABLESPACE TS_session_1;

-- ASSIGNING TABLESPACE TO SESSION_2
ALTER USER session_2  DEFAULT TABLESPACE TS_session_2;
```

```
==========================================================================
```

--Business Intelligence : SESSION_2

```
==========================================================================
```

--We create a PL / SQL procedure that receives two dates and supplies the fact table from
--operational for all orders made between two dates.

```
CREATE OR REPLACE PROCEDURE disply_data(sdate IN varchar2,edate IN varchar2) AS

   CURSOR C1 IS
      SELECT client_id, product_id,zip_code ,quantity,price
         FROM PURCHASE
            WHERE CAST(time_stamp AS DATE)  Between TO_DATE(sdate,'DD-MM-YYYY')
            AND TO_DATE(edate,'DD-MM-YYYY');

      V_PRICE purchase.price%TYPE;
      V_CUSTID purchase.client_id%TYPE;
      V_PRODUCTID purchase.product_id%TYPE;
      V_QUANTITY PURCHASE.QUANTITY%TYPE;
      V_ZIPCODE PURCHASE.ZIP_CODE%TYPE;
BEGIN
      DBMS_OUTPUT.PUT_LINE('ProductId'||'   '||'CustomerId'||'   '||'Zipcode'||'   '||'Quantity'||'
'||'Price');
      OPEN C1;
         LOOP
          FETCH C1 INTO V_PRODUCTID,V_CUSTID,V_ZIPCODE,V_QUANTITY,V_PRICE;
            EXIT WHEN C1%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(V_PRODUCTID||'   '||V_CUSTID||'   '||V_ZIPCODE||'
'||V_QUANTITY||'   '||V_PRICE);
         END LOOP;
      CLOSE C1;
END;
/

SET SERVEROUTPUT ON
EXEC disply_data(sdate=>'02-Jan-2003',edate=>'02-Dec-2004');


==========================================================================

Ref file:03_DECISIONAL_DB.sql;
```

Brief Information about databases:

Ref file :INFO_ABOUT_DECISIONAL_DATABASE.txt
Ref File :INFO_ABOUT_OPERATIONAL _DATABASE.txt