Report

Laboratory Work 8 - 9

Dmitry Ladutsko

August 16 - 17, 2022

1. Prerequisites Task Information

1.1. Passwords Index

Password Group	Login Name	Password
Operation System	root	“rootadmin”
	oracle	“oracleadmin”
Oracle System	sys	“sysadmin”
	system	“sysadmin”
Oracle Users	All DB users	“%PWD%”

1.2. Folder Paths Index

Path Group	Path Description	Path
Operation System	Oracle RDBMS – BIN	/oracle/app/oracle
	Oracle Inventory	/oracle/app/oraInventory
	Oracle Database Storage	/oracle/oradata
	Oracle Install Directory	/oracle/install
Oracle	ORACLE_BASE	/oracle/app/oracle
	ORACLE_HOME	$ORACLE_BASE/product/11.2
FTP	ftp Incoming Folder	`/ftp/incoming`

2. ETL Extraction - BASIC

2.1. Task 01: Extraction Description

<u>The Main Task</u> is to create chapter in the Solution Concept Document that will explain extraction properties and strategy for your Business Task, according Exit Task for Module 6 – Introduction to DWH.

<u>Task Results:</u>

Create required objects:

- Prepare Document with new Chapter.

Note. From my point of view the **offline extraction** can be assessed as followed: as data is not directly taken from the source, it already has an existing structure, namely it can be tablespaces and logs or in another case the structure may be a result of the extraction method.

Another type of extraction, namely, **full extraction** may be described as complete pulling of data directly from the source. This may be beneficial in a sense that there is no need to track the source system as the extraction process reflects all the available data in the system.

Note. New Chapter "Extraction Description" will be added to Business task solution concept

3. ETL Transportation – Example of Loading FCT_*

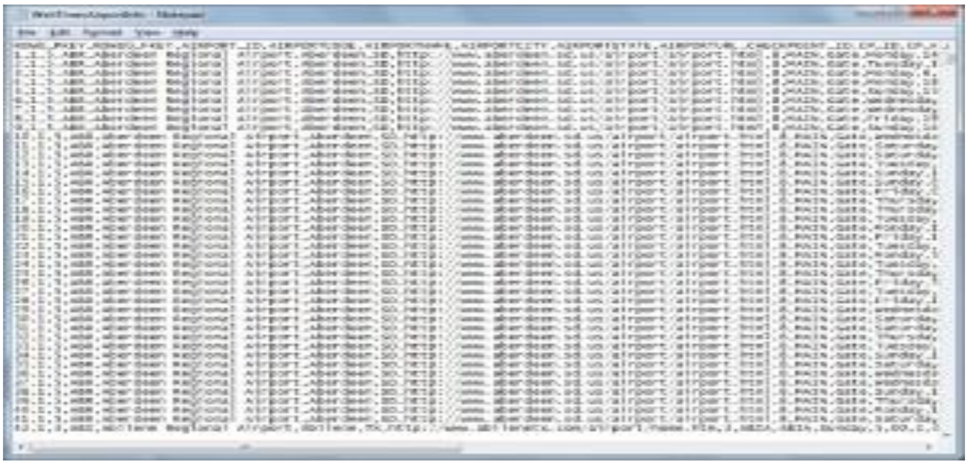
Task 01: Transportation Description

Transportation Using Flat Files

The most common method for transporting data is by the transfer of flat files, using mechanisms such as FTP or other remote file system access protocols. Data is unloaded or exported from the source system into flat files, and is then transported to the target platform using FTP or similar mechanisms.

Source systems and DWH often use different operating systems and database systems, using flat files is often the simplest way to exchange data between heterogeneous systems with minimal transformations. However, even when transporting data between homogeneous systems, flat files are often the most efficient and most easy-to-manage mechanism for data transfer.

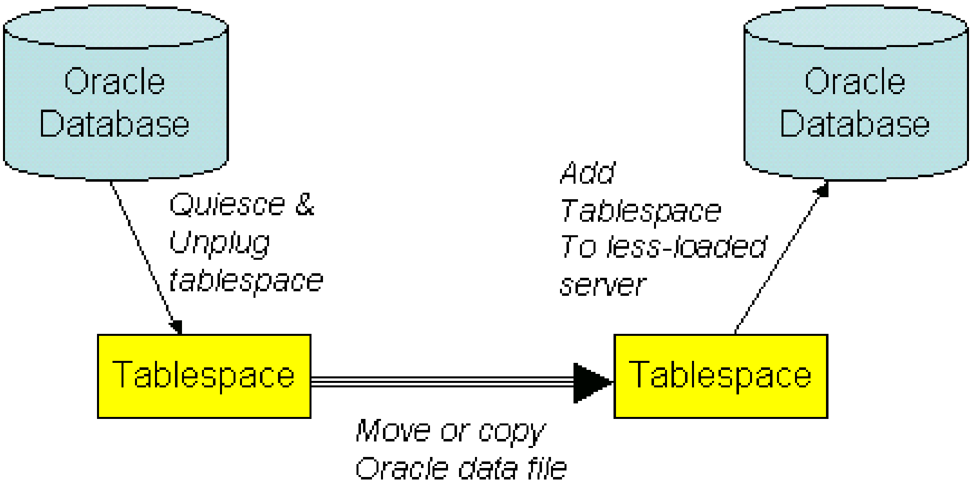
In my business, I guess the best way is working with flat files cause they are easy to generate, use quite a little memory, easy to parse and easy to generate from back – end algorithms.



Transportation Using Transportable Tablespaces

Transportable tablespaces is a feature of the Oracle database, introduced with Oracle 8i, that allows DBAs to copy or move tablespaces between databases. It is especially helpful when tablespace size is in huge, in Terrabytes and tradition ways are not as quicker as scp..ing the datafiles. In other words, Transportable Tablespaces allow Oracle data files to be unplugged from a database and copied to another Oracle instance (on another server) and then added into that Oracle instance (refer to Figure below).

The most common applications of transportable tablespaces in data warehouses are in moving data from a staging database to a data warehouse, or in moving data from a data warehouse to a data mart.


Oracle Database
Quiesce & Unplug tablespace
Tablespace
Move or copy Oracle data file
Tablespace
Add Tablespace To less-loaded server
Oracle Database

Task 02 is common for LabWork 08, 09.

Task 02: Prepare Table of Facts to DW Layer

The Main Task is to create required objects on DW layers and realize load process for Facts

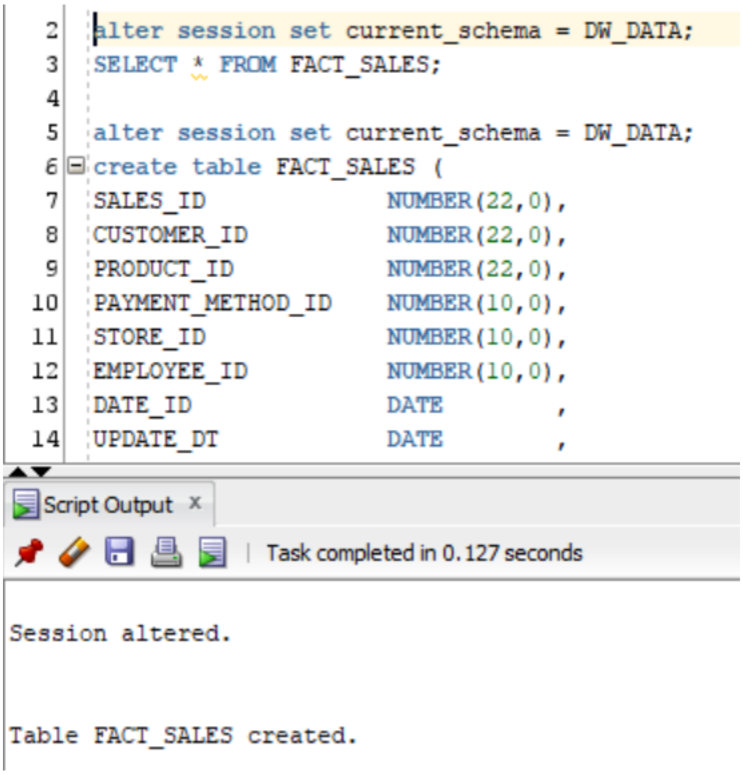
Required points:

- Create new package for Load FCT_* to DW Layer

Task Results:

Create required objects:

- Put objects script to Git.
- Prepare Document with Screenshot of Data

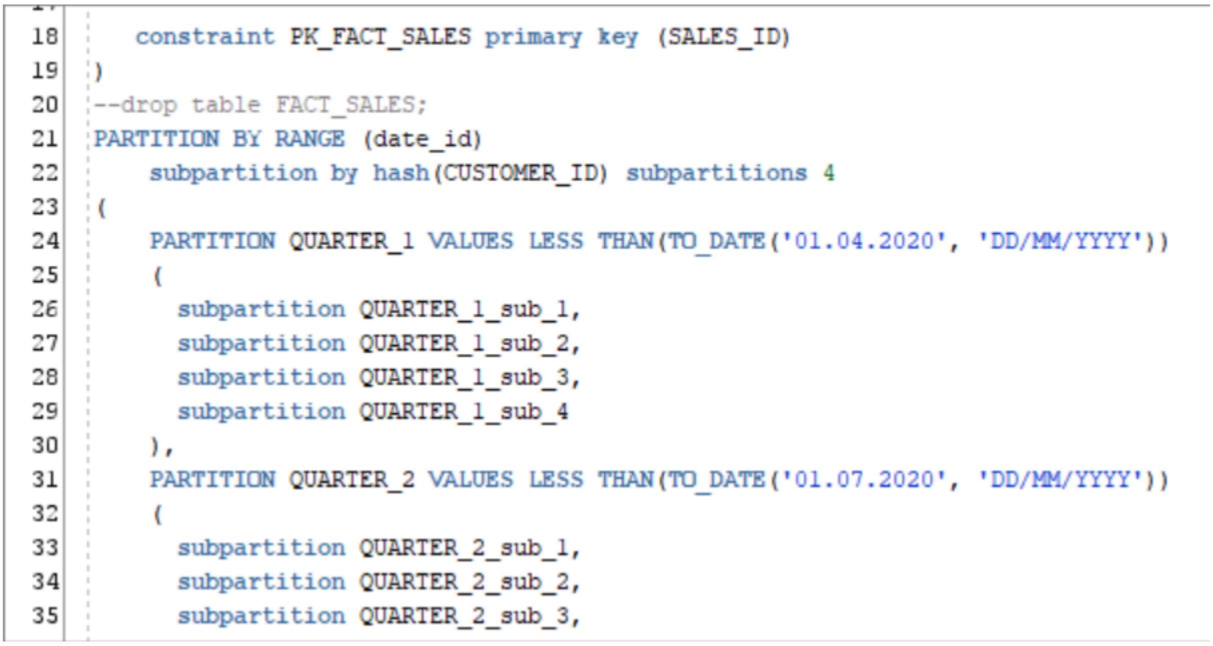


```
alter session set current_schema = DW_DATA;
SELECT * FROM FACT_SALES;

alter session set current_schema = DW_DATA;
create table FACT_SALES (
SALES_ID                  NUMBER(22,0),
CUSTOMER_ID               NUMBER(22,0),
PRODUCT_ID                NUMBER(22,0),
PAYMENT_METHOD_ID         NUMBER(10,0),
STORE_ID                  NUMBER(10,0),
EMPLOYEE_ID               NUMBER(10,0),
DATE_ID                   DATE        ,
UPDATE_DT                 DATE        ,
```

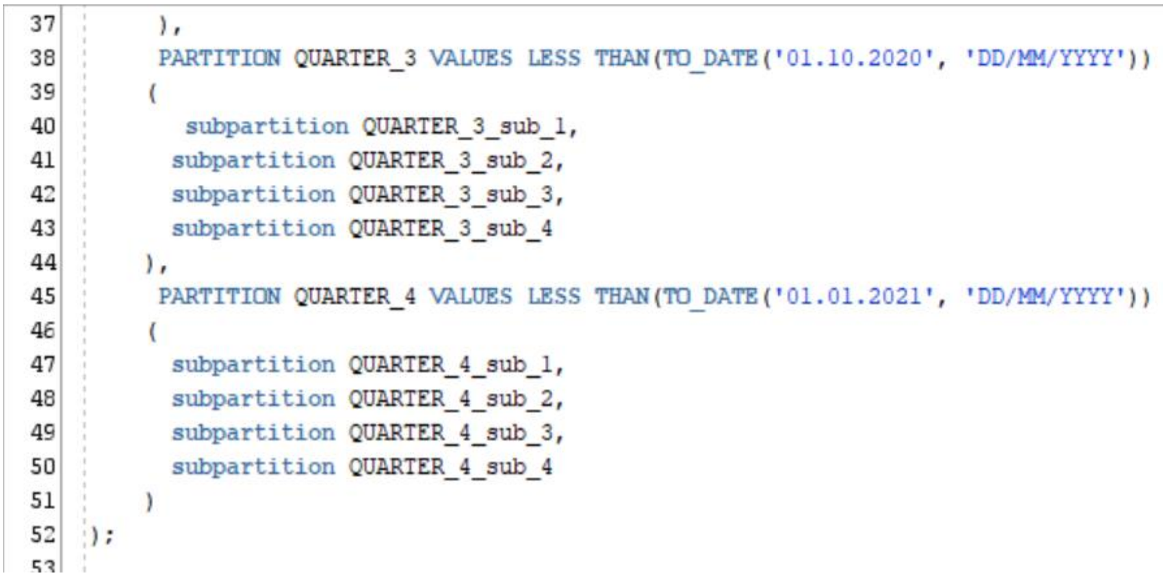
Task completed in 0.127 seconds

```
Session altered.


Table FACT_SALES created.
```

Note. Fact table created

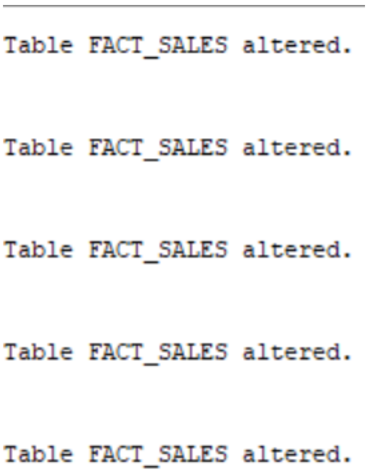


```
   constraint PK_FACT_SALES primary key (SALES_ID)
)
--drop table FACT_SALES;
PARTITION BY RANGE (date_id)
    subpartition by hash(CUSTOMER_ID) subpartitions 4
(
    PARTITION QUARTER_1 VALUES LESS THAN(TO_DATE('01.04.2020', 'DD/MM/YYYY'))
    (
      subpartition QUARTER_1_sub_1,
      subpartition QUARTER_1_sub_2,
      subpartition QUARTER_1_sub_3,
      subpartition QUARTER_1_sub_4
    ),
    PARTITION QUARTER_2 VALUES LESS THAN(TO_DATE('01.07.2020', 'DD/MM/YYYY'))
    (
      subpartition QUARTER_2_sub_1,
      subpartition QUARTER_2_sub_2,
      subpartition QUARTER_2_sub_3,
```

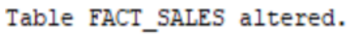


```
     ),
     PARTITION QUARTER_3 VALUES LESS THAN(TO_DATE('01.10.2020', 'DD/MM/YYYY'))
    (
       subpartition QUARTER_3_sub_1,
      subpartition QUARTER_3_sub_2,
      subpartition QUARTER_3_sub_3,
      subpartition QUARTER_3_sub_4
    ),
     PARTITION QUARTER_4 VALUES LESS THAN(TO_DATE('01.01.2021', 'DD/MM/YYYY'))
    (
      subpartition QUARTER_4_sub_1,
      subpartition QUARTER_4_sub_2,
      subpartition QUARTER_4_sub_3,
      subpartition QUARTER_4_sub_4
    )
);
```

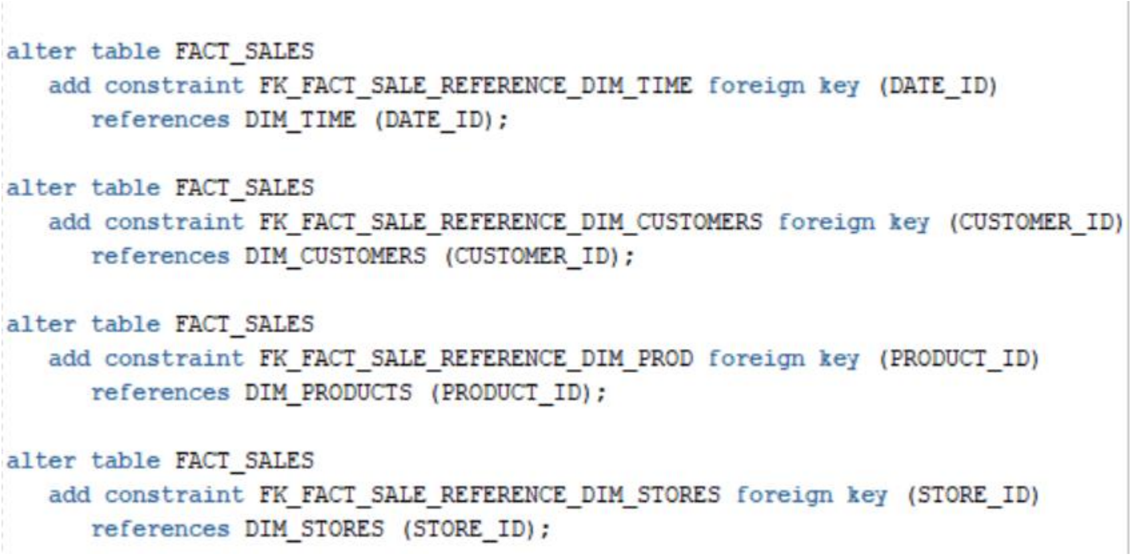
Note. It’s Primary key and partitions (with sub – partitions)



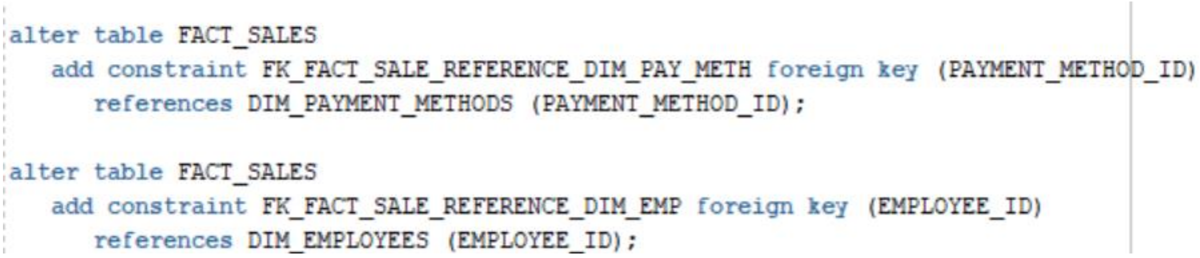
```
Table FACT_SALES altered.


Table FACT_SALES altered.


Table FACT_SALES altered.


Table FACT_SALES altered.


Table FACT_SALES altered.
```



```
Table FACT_SALES altered.
```



```
alter table FACT_SALES
   add constraint FK_FACT_SALE_REFERENCE_DIM_TIME foreign key (DATE_ID)
      references DIM_TIME (DATE_ID);

alter table FACT_SALES
   add constraint FK_FACT_SALE_REFERENCE_DIM_CUSTOMERS foreign key (CUSTOMER_ID)
      references DIM_CUSTOMERS (CUSTOMER_ID);

alter table FACT_SALES
   add constraint FK_FACT_SALE_REFERENCE_DIM_PROD foreign key (PRODUCT_ID)
      references DIM_PRODUCTS (PRODUCT_ID);

alter table FACT_SALES
   add constraint FK_FACT_SALE_REFERENCE_DIM_STORES foreign key (STORE_ID)
      references DIM_STORES (STORE_ID);
```

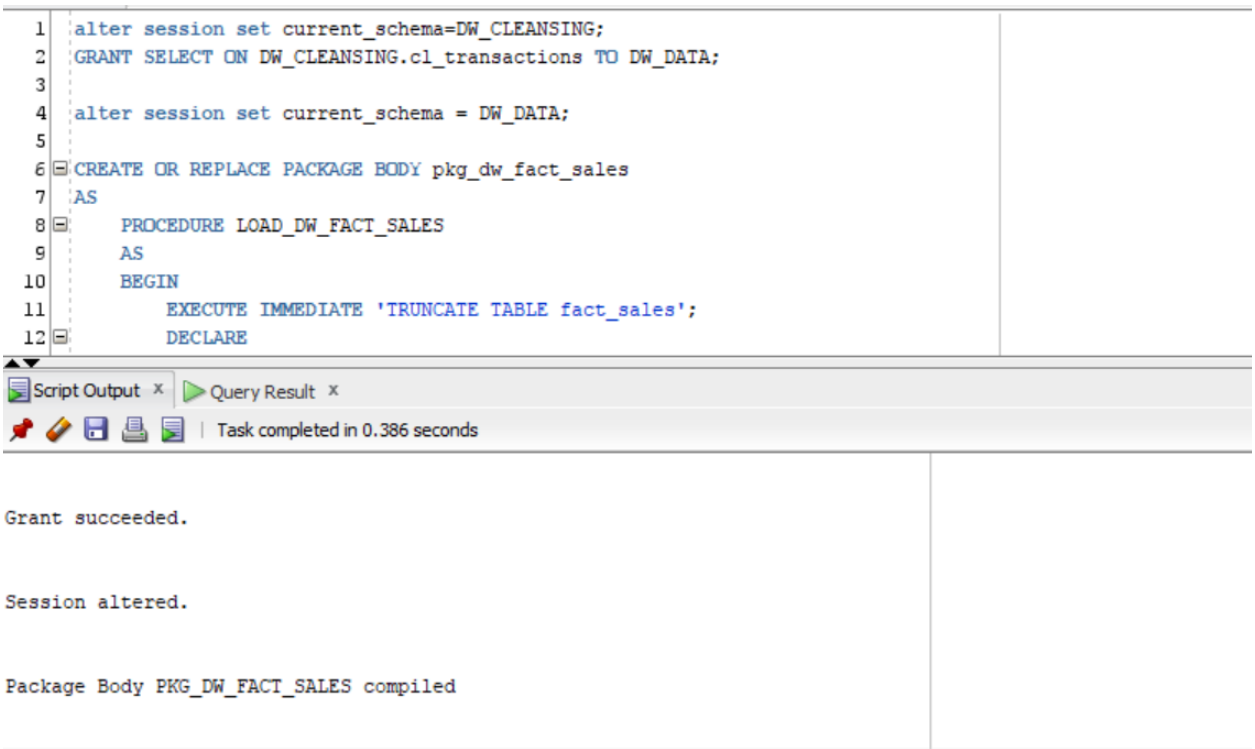


```
alter table FACT_SALES
   add constraint FK_FACT_SALE_REFERENCE_DIM_PAY_METH foreign key (PAYMENT_METHOD_ID)
      references DIM_PAYMENT_METHODS (PAYMENT_METHOD_ID);

alter table FACT_SALES
   add constraint FK_FACT_SALE_REFERENCE_DIM_EMP foreign key (EMPLOYEE_ID)
      references DIM_EMPLOYEES (EMPLOYEE_ID);
```

Note. Foreign key constraints. FACT_SALES TABLE CREATED BEFORE, IN MODULE 1, but now a little bit modified. DDL script added to Laboratory Work 8 – 9 folder and exit task folder.

Note. I created DDL script to create **table of facts** as you have seen. Now I created packages to load identificators into Fact Table.

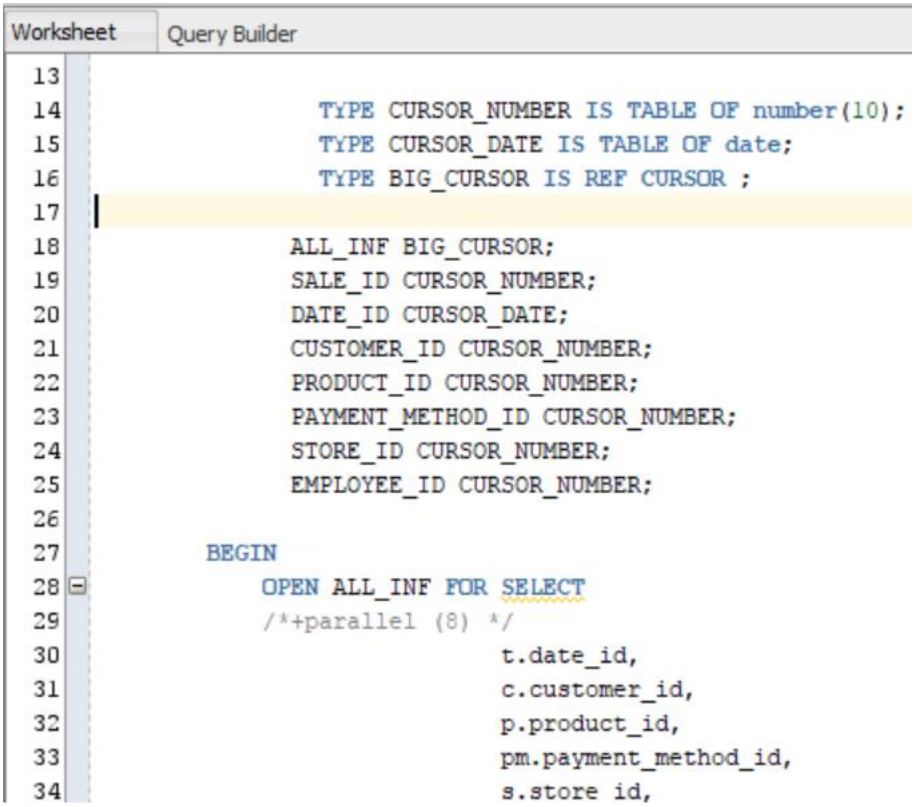
Note. Table of facts may contain other data such as some simple aggregations, but I left it in conservative way, anyway simple aggregations will be easily added later (or use Data Marts for complex reporting). Anyways I will add it :)



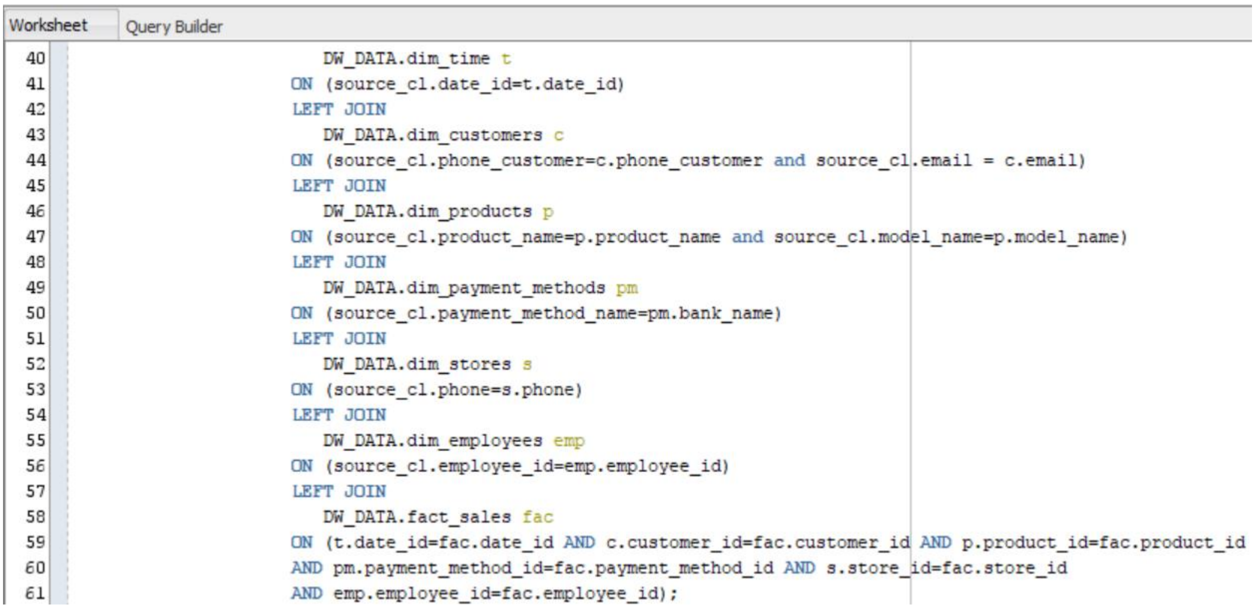
```
alter session set current_schema=DW_CLEANSING;
GRANT SELECT ON DW_CLEANSING.cl_transactions TO DW_DATA;

alter session set current_schema = DW_DATA;

CREATE OR REPLACE PACKAGE BODY pkg_dw_fact_sales
AS
    PROCEDURE LOAD_DW_FACT_SALES
    AS
    BEGIN
        EXECUTE IMMEDIATE 'TRUNCATE TABLE fact_sales';
        DECLARE
```

Task completed in 0.386 seconds

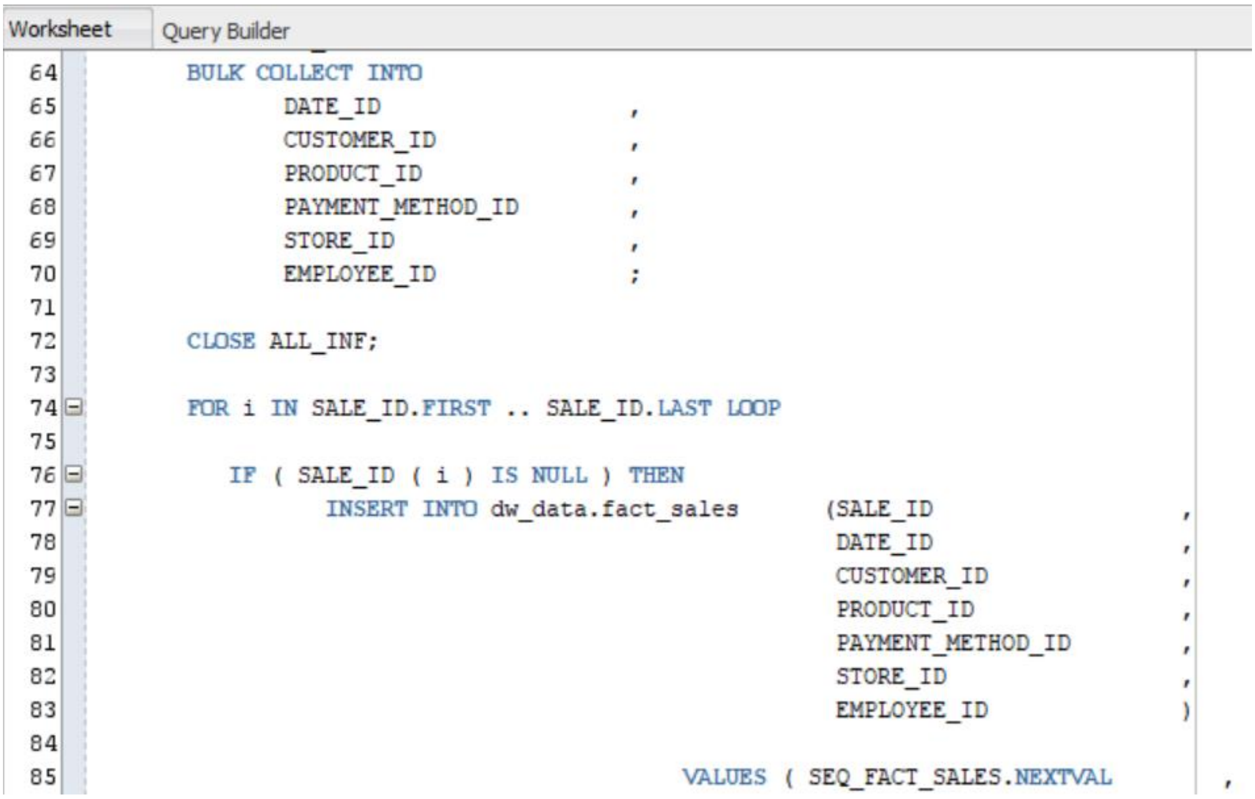
```
Grant succeeded.


Session altered.


Package Body PKG_DW_FACT_SALES compiled
```



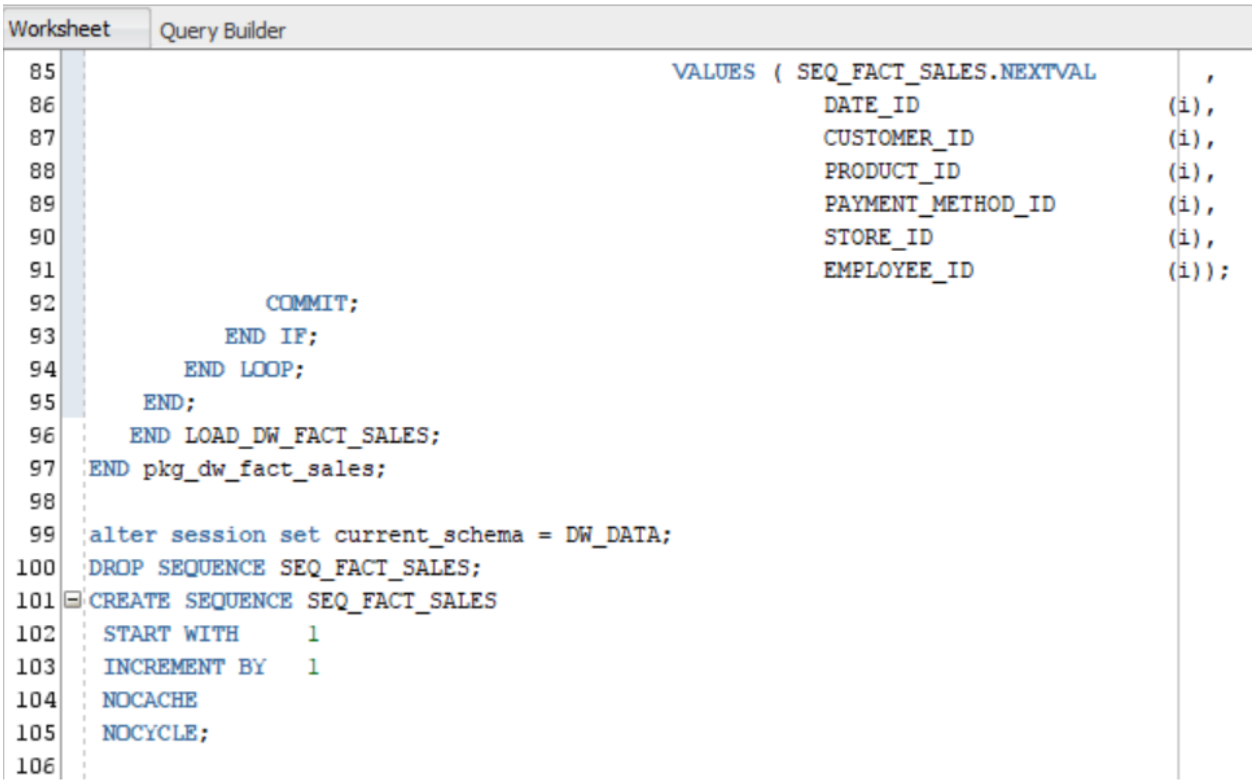
```

                TYPE CURSOR_NUMBER IS TABLE OF number(10);
                TYPE CURSOR_DATE IS TABLE OF date;
                TYPE BIG_CURSOR IS REF CURSOR ;

              ALL_INF BIG_CURSOR;
              SALE_ID CURSOR_NUMBER;
              DATE_ID CURSOR_DATE;
              CUSTOMER_ID CURSOR_NUMBER;
              PRODUCT_ID CURSOR_NUMBER;
              PAYMENT_METHOD_ID CURSOR_NUMBER;
              STORE_ID CURSOR_NUMBER;
              EMPLOYEE_ID CURSOR_NUMBER;

        BEGIN
            OPEN ALL_INF FOR SELECT
            /*+parallel (8) */
                             t.date_id,
                             c.customer_id,
                             p.product_id,
                             pm.payment_method_id,
                             s.store_id,
```



```
                        DW_DATA.dim_time t
                     ON (source_cl.date_id=t.date_id)
                     LEFT JOIN
                        DW_DATA.dim_customers c
                     ON (source_cl.phone_customer=c.phone_customer and source_cl.email = c.email)
                     LEFT JOIN
                        DW_DATA.dim_products p
                     ON (source_cl.product_name=p.product_name and source_cl.model_name=p.model_name)
                     LEFT JOIN
                        DW_DATA.dim_payment_methods pm
                     ON (source_cl.payment_method_name=pm.bank_name)
                     LEFT JOIN
                        DW_DATA.dim_stores s
                     ON (source_cl.phone=s.phone)
                     LEFT JOIN
                        DW_DATA.dim_employees emp
                     ON (source_cl.employee_id=emp.employee_id)
                     LEFT JOIN
                        DW_DATA.fact_sales fac
                     ON (t.date_id=fac.date_id AND c.customer_id=fac.customer_id AND p.product_id=fac.product_id
                     AND pm.payment_method_id=fac.payment_method_id AND s.store_id=fac.store_id
                     AND emp.employee_id=fac.employee_id);
```



```
       BULK COLLECT INTO
              DATE_ID                  ,
              CUSTOMER_ID              ,
              PRODUCT_ID               ,
              PAYMENT_METHOD_ID        ,
              STORE_ID                 ,
              EMPLOYEE_ID              ;

       CLOSE ALL_INF;

       FOR i IN SALE_ID.FIRST .. SALE_ID.LAST LOOP

          IF ( SALE_ID ( i ) IS NULL ) THEN
                 INSERT INTO dw_data.fact_sales      (SALE_ID                 ,
                                                      DATE_ID                 ,
                                                      CUSTOMER_ID             ,
                                                      PRODUCT_ID              ,
                                                      PAYMENT_METHOD_ID       ,
                                                      STORE_ID                ,
                                                      EMPLOYEE_ID             )

                                           VALUES ( SEQ_FACT_SALES.NEXTVAL      ,
```



```
                                          VALUES ( SEQ_FACT_SALES.NEXTVAL       ,
                                                      DATE_ID                  (i),
                                                      CUSTOMER_ID              (i),
                                                      PRODUCT_ID               (i),
                                                      PAYMENT_METHOD_ID        (i),
                                                      STORE_ID                 (i),
                                                      EMPLOYEE_ID              (i));
             COMMIT;
          END IF;
       END LOOP;
    END;
   END LOAD_DW_FACT_SALES;
END pkg_dw_fact_sales;

alter session set current_schema = DW_DATA;
DROP SEQUENCE SEQ_FACT_SALES;
CREATE SEQUENCE SEQ_FACT_SALES
 START WITH     1
 INCREMENT BY   1
 NOCACHE
 NOCYCLE;
```

Note. I used **Procedure with Cursor** to move id's.

Laboratory Work Summary

At this Laboratory Work we practiced more how to create fact table, reference it with dimensions, use partitions and sub – partitions etc.

We learned more about Transportation and Extraction principals and about their types. Learned such types of architectures as Lambda and Kappa One more time used Cursors, so also practised more and more.