Report

Laboratory Work 10

Dmitry Ladutsko

August 18, 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 Transformation - BASIC

2.1. Task 01: Transformation Description

The Main Task is to create chapter in the Solution Concept Document that will explain Transformation strategy for your Business Task, according Exit Task for Module 6 – Introduction to DWH.

SQL Transformation

The SQL transformation processes SQL queries midstream in a pipeline. The SQL transformation can be an active or passive transformation. You can insert, delete, update, and retrieve rows from a database. You can pass the database connection information to the SQL transformation as input data at run time. The transformation processes external SQL scripts or SQL queries that you create in an SQL editor. The SQL transformation processes the query and returns rows and database errors.

PL/SQL Transforming

In a data warehouse environment, you can use procedural languages such as PL/SQL to implement complex transformations in the Oracle Database. Whereas CTAS operates on entire tables and emphasizes parallelism, PL/SQL provides a row-based approached and can accommodate very sophisticated transformation rules.

For example, a PL/SQL procedure could open multiple cursors and read data from multiple source tables, combine this data using complex business rules, and finally insert the transformed data into one or more target table. It would be difficult or impossible to express the same sequence of operations using standard SQL statements.

Transforming Data Using Table Functions

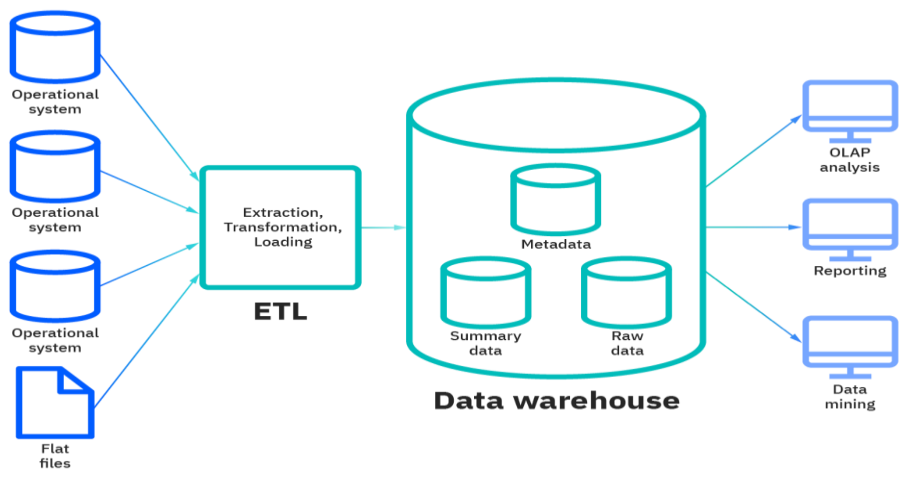
Table functions provide the support for pipelined and parallel execution of transformations implemented in PL/SQL, C, or Java.

Scenarios as mentioned earlier can be done without requiring the use of intermediate staging tables, which interrupt the data flow through various transformations steps.

Business solution concept Transformation Strategy

For my business the right way is to use a combo of Transformation types. Thus different objects need to be transformed in different ways.

Reference tables are not needed to be transformed, for example, using PL/SQL Transformation Strategy as well as Customers and Employees Dimensions and Fact Sales table, for example are needed to be updated or loaded within new data using Multiple Cursors, Functions and Procedures to manipulate with several objects at the same time.



3. ETL Transformation – Loading SAL

Task 01 is common for LabWork 10 (Task 02), 11 (Task 01).

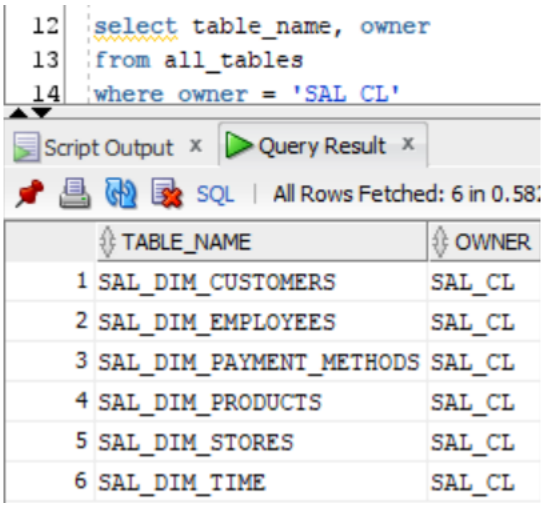
3.1. Task 02: Loading to SAL Layer Data

The Main Task is to load dimension to SAL layer

Required points:

- Create new package for Load FCT_* and DIM_* to SAL Layer
- Load Dimension
- Load SCD Dimension
- Load FCT_*

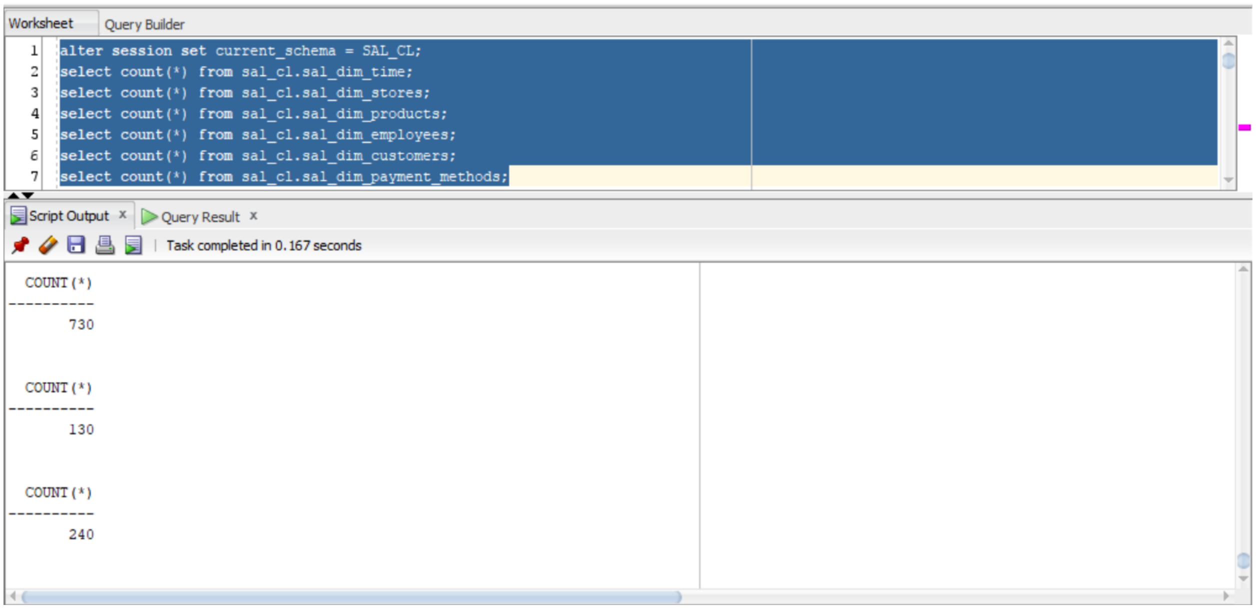
Note. Let's create tables on SAL Layer using created before user SAL_CL (all created tables list):



```
select table_name, owner
from all_tables
where owner = 'SAL_CL'
```

	TABLE_NAME	OWNER
1	SAL_DIM_CUSTOMERS	SAL_CL
2	SAL_DIM_EMPLOYEES	SAL_CL
3	SAL_DIM_PAYMENT_METHODS	SAL_CL
4	SAL_DIM_PRODUCTS	SAL_CL
5	SAL_DIM_STORES	SAL_CL
6	SAL_DIM_TIME	SAL_CL

Note. Now we need to move data **from DW layer to SAL layer**. I used procedures with cursor for it. I've also created **references** between created tables to later easily **join** them using in **incapsulated into views reports.**



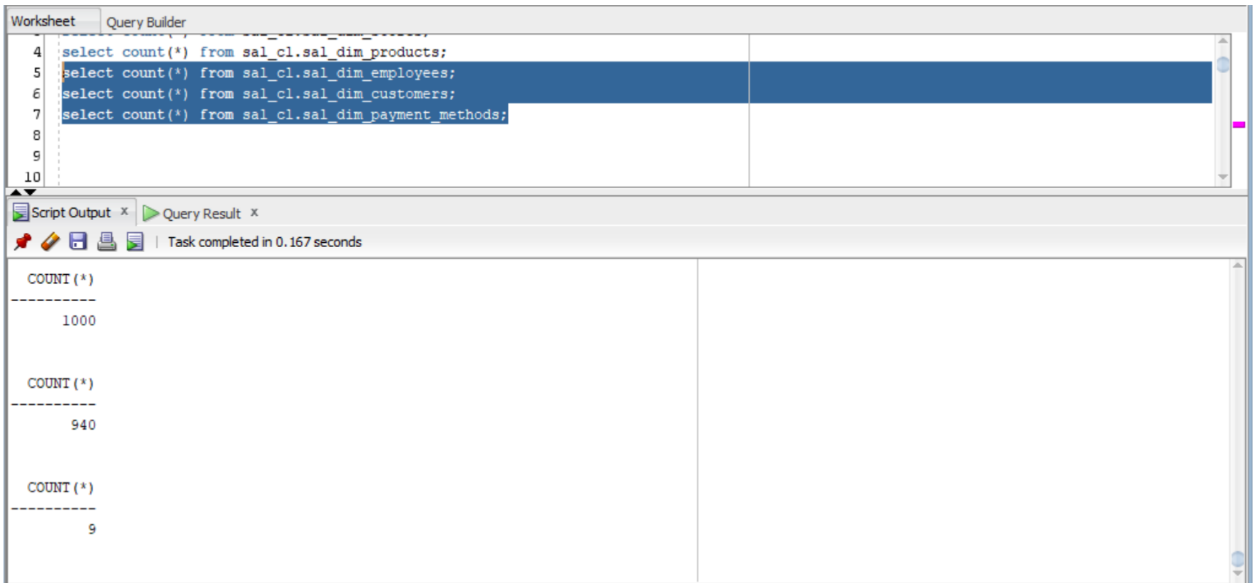
Worksheet Query Builder

```
alter session set current_schema = SAL_CL;
select count(*) from sal_cl.sal_dim_time;
select count(*) from sal_cl.sal_dim_stores;
select count(*) from sal_cl.sal_dim_products;
select count(*) from sal_cl.sal_dim_employees;
select count(*) from sal_cl.sal_dim_customers;
select count(*) from sal_cl.sal_dim_payment_methods;
```

Script Output × Query Result ×

Task completed in 0.167 seconds

```
  COUNT(*)
----------
       730


  COUNT(*)
----------
       130


  COUNT(*)
----------
       240
```



Worksheet Query Builder

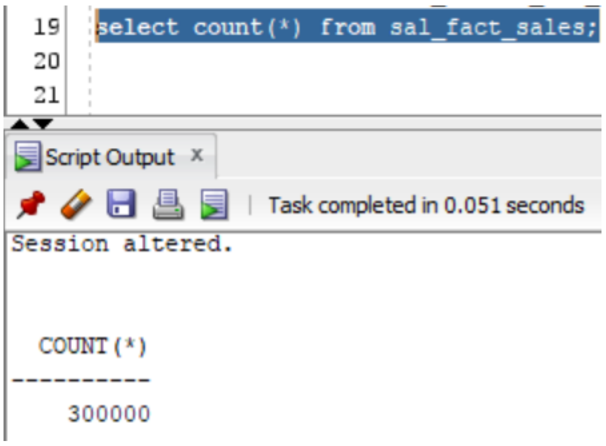
```
select count(*) from sal_cl.sal_dim_products;
select count(*) from sal_cl.sal_dim_employees;
select count(*) from sal_cl.sal_dim_customers;
select count(*) from sal_cl.sal_dim_payment_methods;
```

Script Output × Query Result ×

Task completed in 0.167 seconds

```
  COUNT(*)
----------
      1000


  COUNT(*)
----------
       940


  COUNT(*)
----------
         9
```

Note. As you can see at the screenshot above, all tables are fulfilled with data from previous layer(DW). So, now we need to make the same step to create **fact table on SAL level** and **fulfill it with data**

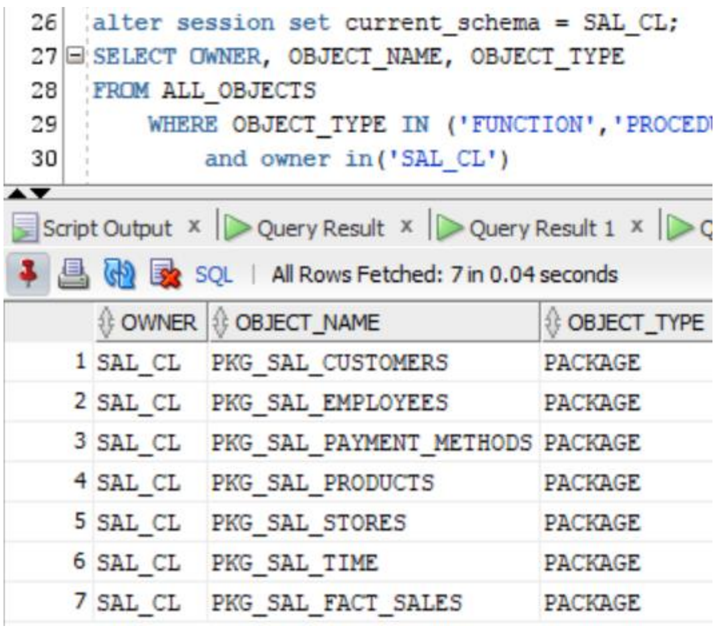


```
select count(*) from sal_fact_sales;
```

Script Output ×

Task completed in 0.051 seconds

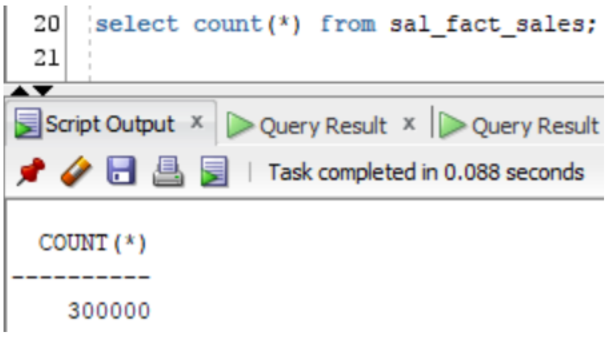
```
Session altered.


  COUNT(*)
----------
    300000
```



```
alter session set current_schema = SAL_CL;
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM ALL_OBJECTS
    WHERE OBJECT_TYPE IN ('FUNCTION','PROCED
        and owner in('SAL_CL')
```

All Rows Fetched: 7 in 0.04 seconds

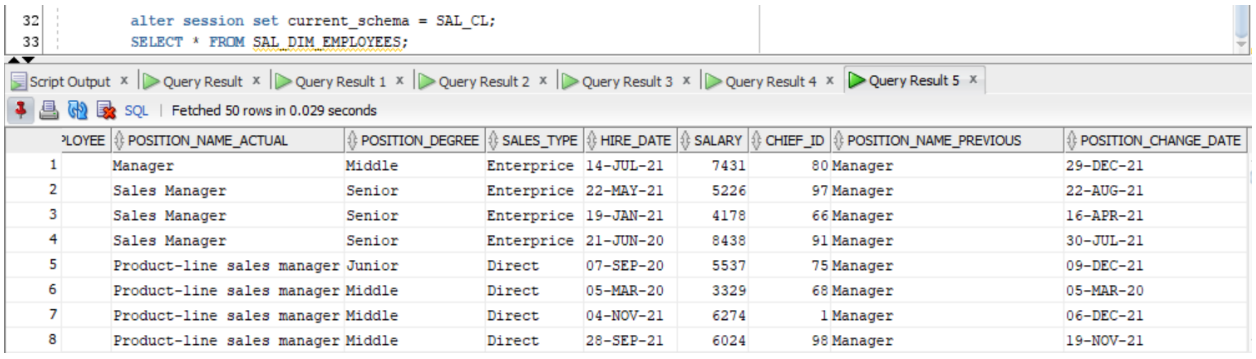
	OWNER	OBJECT_NAME	OBJECT_TYPE
1	SAL_CL	PKG_SAL_CUSTOMERS	PACKAGE
2	SAL_CL	PKG_SAL_EMPLOYEES	PACKAGE
3	SAL_CL	PKG_SAL_PAYMENT_METHODS	PACKAGE
4	SAL_CL	PKG_SAL_PRODUCTS	PACKAGE
5	SAL_CL	PKG_SAL_STORES	PACKAGE
6	SAL_CL	PKG_SAL_TIME	PACKAGE
7	SAL_CL	PKG_SAL_FACT_SALES	PACKAGE



```
select count(*) from sal_fact_sales;
```

Task completed in 0.088 seconds

```
 COUNT(*)
----------
    300000
```



```
alter session set current_schema = SAL_CL;
SELECT * FROM SAL_DIM_EMPLOYEES;
```

Fetched 50 rows in 0.029 seconds

	PLOYEE	POSITION_NAME_ACTUAL	POSITION_DEGREE	SALES_TYPE	HIRE_DATE	SALARY	CHIEF_ID	POSITION_NAME_PREVIOUS	POSITION_CHANGE_DATE
1		Manager	Middle	Enterprice	14-JUL-21	7431	80	Manager	29-DEC-21
2		Sales Manager	Senior	Enterprice	22-MAY-21	5226	97	Manager	22-AUG-21
3		Sales Manager	Senior	Enterprice	19-JAN-21	4178	66	Manager	16-APR-21
4		Sales Manager	Senior	Enterprice	21-JUN-20	8438	91	Manager	30-JUL-21
5		Product-line sales manager	Junior	Direct	07-SEP-20	5537	75	Manager	09-DEC-21
6		Product-line sales manager	Middle	Direct	05-MAR-20	3329	68	Manager	05-MAR-20
7		Product-line sales manager	Middle	Direct	04-NOV-21	6274	1	Manager	06-DEC-21
8		Product-line sales manager	Middle	Direct	28-SEP-21	6024	98	Manager	19-NOV-21

Note. Loaded SCD type 3 table (*_employees)

```
alter session set current_schema = SAL_CL;
begin
pkg_sal_time.LOAD_SAL_TIME;
pkg_sal_stores.LOAD_SAL_STORES;
pkg_sal_products.LOAD_SAL_PRODUCTS;
pkg_sal_customers.LOAD_SAL_CUSTOMERS;
pkg_sal_employees.LOAD_SAL_EMPLOYEES;
pkg_sal_payment_methods.LOAD_SAL_PAYMENT_METHODS;
pkg_sal_fact_sales.LOAD_SAL_FACT_SALES;
end;
```

Note. I have summered all packages in one procedure and also putted it on PACKAGES folder.

Also will do the same with other layers load (SA - > CL - > DW - > SAL). File is going to be named as **full load package (*).sql**

Note. (e.g. full load package DW-SAL.sql)

NOTE. New **views with group** statements will be created in **Laboratory Work 11**.

Laboratory Work Summary

At this Laboratory Work we practiced more about how to create **SAL Layer**, how faster load data from previous layers. How faster we could load it.

We have learned more about **Transformation** and it types. Used our gotten knowledge to specify and describe our Business Solution Transformation Strategy.