Report

Laboratory Work 4

Dmitry Ladutsko

August 13, 2022

1. Prerequisites

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. Business analyses tasks – Reports

2.1. Task 01: Create Packages for Reload Dimension from SA_*

The Main Task is to independent packages to reload dimension according your DWH solution concept which was developed on Module 6. Introduction to DWH.

Required points:

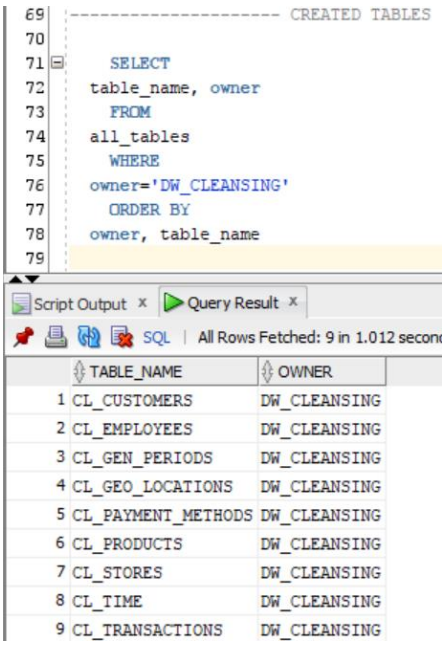
- Create all required Dim objects on DW Layer
- Grant all required Privileges to DW_CL (Cleansing Layer)
- Create Packages to reload Dim data (one package = one dimension.)
- Example future SAL.DIM_GEO_SCD will store all procedure on pkg_etl_dim_geo_dw. But this package will store all small sub dims – T_COUNTRIES, T_REGIONS etc.)
    - Use Explicit Cursor (One package)
    - Use Explicit Cursor and FORALL Bulk Insertion (One package)

- Use Variable Cursor and FORALL Bulk Insertion (One package)
- Use Merge (One packages)

Task Results:

Create required objects:

- Put objects script to Git.
- Prepare Document with Screenshot of Data on Dimensions
- Test data for consistent
- Test Procedure for Repeatable execution (Nothing should change)



Firstly, I created tables on Cleansing level

```
SELECT
        table_name, owner
FROM
        all_tables
WHERE
        owner='DW_CLEANSING'
ORDER BY
        owner, table_name
```

- CL_CUSTOMERS
- CL_EMPLOYEES
- CL_GEN_PERIODS
- CL_GEO_LOCATIONS
- CL_PAYMENT_METHODS
- CL_PRODUCTS
- CL_STORES
- CL_TIME
- CL_TRANSACTIONS

Note. All scripts stored in GitHub, exit task folder. But also duplicated in lab4 folder AS CL folder.

Then I began **creating packages**, which consist of 2 files :

- *table_name*_define.sql
- *table_name*_body.sql

SQL Worksheet | History

Worksheet | Query Builder

```
--drop package pkg_cl_transaction;
alter session set current_schema=DW_CLEANSING;
SELECT
    *
FROM
    ALL_OBJECTS
WHERE
    OBJECT_TYPE
                IN ('FUNCTION','PROCEDURE','PACKAGE')
                AND OWNER = 'DW_CLEANSING'
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

SQL | All Rows Fetched: 8 in 0.149 seconds

	OWNER	OBJECT_NAME	SUBOBJECT_NAME	OBJECT_ID	DATA_OBJECT_ID	OBJECT_TYPE
1	DW_CLEANSING	PKG_CL_CUSTOMERS	(null)	70497	(null)	PACKAGE
2	DW_CLEANSING	PKG_CL_PAYMENT_METHODS	(null)	70499	(null)	PACKAGE
3	DW_CLEANSING	PKG_CL_LOCATIONS	(null)	70500	(null)	PACKAGE
4	DW_CLEANSING	PKG_CL_PERIODS	(null)	70501	(null)	PACKAGE
5	DW_CLEANSING	PKG_CL_PRODUCTS	(null)	70502	(null)	PACKAGE
6	DW_CLEANSING	PKG_CL_STORES	(null)	70503	(null)	PACKAGE
7	DW_CLEANSING	PKG_CL_TIME	(null)	70504	(null)	PACKAGE
8	DW_CLEANSING	PKG_CL_TRANSACTION	(null)	70505	(null)	PACKAGE

Note. Later I will show the **Data Flow Diagram** to explain how I store files / sub – folders / folders

Now let's try to create package bodies. And of course execute them to move data from SA level to CL level

```
ALTER USER DW_CLEANSING quota unlimited on TS_DW_CLEANSING;
alter session set current_schema=DW_CLEANSING;
exec pkg_cl_customers.LOAD_CLEAN_CUSTOMERS;
```

Script Output | Query Result

Task completed in 0.145 seconds

```
*Action:   Grant the user the appropriate system privileges or grant the user
           space resource on the tablespace.

User DW_CLEANSING altered.


Session altered.


PL/SQL procedure successfully completed.
```

***After almost an hour** I finally understood that it is a great idea to grant my user more space resource :)

```
70  alter session set current_schema = SA_CUSTOMERS;
71  select count (*) from sa_customers
72  commit;
```

Script Output | Query Result

Task completed in 0.07 seconds

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

```
69  alter session set current_schema=DW_CLEANSING;
70  select count (*) from cl_customers;
71
72
```

Script Output | Query Result

Task completed in 0.059 seconds

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

SQL Worksheet | History

0.076 seconds

Worksheet | Query Builder

```
81
82  alter session set current_schema=DW_CLEANSING;
83  select count(*) from cl_employees;
84
85  alter session set current_schema=SA_PRODUCTS;
86  select count(*) from SA_employees;
87
88  commit;
```

Script Output

Task completed in 0.076 seconds

```
  COUNT(*)
----------
      1500


Session altered.


  COUNT(*)
----------
      1500
```

As you can see, if we **count** rows from the same tables (but on different layers(different tablespaces)) thee number of rows differs because in *.SA table were some rows with NULL values.

Note. Next I am going to make a body statements for every table (except of some(1 – 3), which are not needed to be cleansed)

Note. However, employees table **was not cleansed** because of they were created using ***NOT NULL* constraint**. It seems to be useless cleansing such

tables but if someone will change this table structure **in future** and try to use *null* values in some rows, then **it will be worthy**.

Note. I coded every package and executed their bodies before moving data from sa_* tables to cl_* tables, and generating transactions table on cleansing layer.

```
SQL Worksheet  History
Worksheet  Query Builder
66  FROM
67     DW_CLEANSING.cl_products p
68
69     CROSS JOIN DW_CLEANSING.cl_customers c
70
71         CROSS JOIN DW_CLEANSING.cl_payment_methods pm
72
73             CROSS JOIN DW_CLEANSING.CL_EMPLOYEES e
74
75             INNER JOIN DW_CLEANSING.cl_time t ON t.date_id = c.CUSTOMER_SALE_DATE
76
77  WHERE c.CUSTOMER_SALE_DATE > TO_DATE( '01.01.20', 'MM/DD/YY' )
78         AND c.CUSTOMER_SALE_DATE < TO_DATE( '01.01.22', 'MM/DD/YY' ) and
79
80         (mod(c.PHONE_CUSTOMER,2) = 0 ) and
81
82         e.phone_EMPLOYEE LIKE '2%' AND
83
84         e.FIRST_NAME_EMPLOYEE LIKE 'D%'
85                                         );
Script Output  Query Result  Query Result 1
Task completed in 15.367 seconds
2,436,480 rows inserted.
```

Note. Finally, I got ~2.5 m rows inserted into cl.transaction table fulfilled with fully cleansed data (of course with some artificial constraints)!

Note. In fact, we need to add only decision – making data. I thought that I could throw away some columns (e.g. customers/employees emails, phones etc.), but then I realised I could use it to, for example, select customers emails to target marketing content. So, anyway if I decide it is not needed, I will alter this tables.

dim_customers.sql	28-Jul-22 1:47 PM	SQL File
dim_employees.sql	27-Jul-22 8:36 AM	SQL File
dim_gen_periods.sql	27-Jul-22 7:43 AM	SQL File
dim_geo_location.sql	27-Jul-22 7:43 AM	SQL File
dim_payment_methods.sql	27-Jul-22 8:36 AM	SQL File
dim_products.sql	27-Jul-22 8:36 AM	SQL File
dim_store.sql	28-Jul-22 1:47 PM	SQL File
dim_time.sql	28-Jul-22 1:47 PM	SQL File

Note. We have created dimension tables before. Copy of dimensions creation scripts duplicated in lab4 folder.

```
DROP SEQUENCE SEQ_CUSTOMERS;
CREATE SEQUENCE SEQ_CUSTOMERS
 START WITH      1
 INCREMENT BY    1
 NOCACHE
 NOCYCLE;
```

Note. I created such sequences to auto increment primary key id's. Then again added packages using following structure :

- *table_name*_define.sql
- *table_name*_body.sql

```
Worksheet   Query Builder

CREATE OR REPLACE PACKAGE body pkg_dw_customers
 AS
   PROCEDURE LOAD_DW_CUSTOMERS
    AS
      BEGIN
      MERGE INTO DW_DATA.dim_customers A
      USING ( SELECT FIRST_NAME_CUSTOMER, LAST_NAME_CUSTOMER,COUNTRY_CITY_CUSTOMER, ADRESS_CUSTOMER, EMAIL, PHONE_CUSTOMER, AGE,IS_ACTI
              FROM DW_CLEANSING.cl_customers) B
              ON (a.PHONE_CUSTOMER = b.PHONE_CUSTOMER)
              WHEN MATCHED THEN
                 UPDATE SET a.ADRESS_CUSTOMER = b.ADRESS_CUSTOMER
              WHEN NOT MATCHED THEN
                 INSERT (a.CUSTOMER_ID, a.FIRST_NAME_CUSTOMER, a.LAST_NAME_CUSTOMER,a.COUNTRY_CITY_CUSTOMER, a.ADRESS_CUSTOMER, a.EMAIL
                 VALUES (SEQ_CUSTOMERS.NEXTVAL, b.FIRST_NAME_CUSTOMER, b.LAST_NAME_CUSTOMER,b.COUNTRY_CITY_CUSTOMER, b.ADRESS_CUSTOMER,
      COMMIT;
    END LOAD_DW_CUSTOMERS;
 END pkg_dw_customers;
```

Script Output × | Query Result ×

SQL | Fetched 50 rows in 0.038 seconds

	CUSTOMER_ID	FIRST_N...	LAST_NAME...	COUNTRY_CITY_C...	ADRESS_CUSTOMER	EMAIL	PHONE_CUSTOMER	AGE	IS_ACTIVE	CUSTOMER_SALE_DATE
1	1	Nikita...	Smith ...	Moldova Kishi...	Kosmonavtov st....	meme@g...	2090124	46	Error	31-MAR-21
2	2	Stan ...	Grey ...	UK Manchester...	Independent st....	dmdmd@...	5774337	35	yes	10-MAY-21
3	3	Nikita...	Sobolev ...	USA Californi...	Independent st....	gog@gm...	1102543	49	no	10-MAY-21
4	4	Masha ...	Lee ...	Poland Warsaw...	Independent st....	meme@g...	5768220	52	Error	10-MAY-21
5	5	Nikita...	Smith ...	Moldova Kishi...	Kosmonavtov st....	meme@g...	2090124	46	Error	27-FEB-20
6	6	Masha ...	Lee ...	Poland Warsaw...	Independent st....	meme@g...	5768220	52	Error	08-MAY-20

I used **Use Explicit Cursor** in pkg_cl_employees_body.sql

Merge in pkg_dw_employees_body.sql

Explicit Cursor and FOR ALL BLUNK Insertion in pkg_dw_stores_body.sql

```
52  select * from dim_stores;
53
```

Script Output × | Query Result ×

SQL | Fetched 50 rows in 0.028 seconds

	STORE_ID	COUNTRY_CITY	ADDRESS	PHONE
1	1	Belarus Minsk	Bogdanovicha st.	7313657
2	2	Belarus Minsk	Bogdanovicha st.	3336504
3	3	Belarus Minsk	Bogdanovicha st.	3799744
4	4	Belarus Minsk	Tiger st.	4622645
5	5	Belarus Minsk	Independent st.	6014347
6	6	Belarus Minsk	Park st.	4172971

```
13  PROCEDURE LOAD_DW_STORE
14   AS
15     BEGIN
16      DECLARE
17       TYPE CURSOR_NUMBER IS TABLE OF NUMBER;
18       TYPE CURSOR_VARCHAR IS TABLE OF VARCHAR2(20);
19       TYPE BIG_CURSOR IS REF CURSOR ;
20
21    ALL_INF BIG_CURSOR;
22
23            store_id      CURSOR_NUMBER      ;
24            country_city  CURSOR_VARCHAR     ;
25            address       CURSOR_VARCHAR     ;
26            phone_STAGE CURSOR_NUMBER        ;
27
28  BEGIN
29     OPEN ALL_INF FOR
30         SELECT
31                 source_CL.country_city AS country_city_source_CL,
32                 source_CL.address AS address_source_CL,
33                 stage.phone AS phone,
```

Laboratory Work Summary: At this laboratory work we created Cleansing and Data warehouse layers, created tables for them and practiced how using different methods we can move data from one layer to another. I used following methods for data movements between different layers:

- Explicit Cursor
- Explicit Cursor and FORALL Bulk Insertion
- Variable Cursor and FORALL Bulk Insertion
- Merge