

Report

Laboratory Work 6

Dmitry Ladutsko

August 14, 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. Business analyses tasks – Dimensions

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.

Note. Let's rewrite package which move data from cleansing level (cl_employees) to DW level (dim_employees)

1) Use Execute Immediate with Bind Parameters:

```
PROCEDURE LOAD_DW_EMPLOYEES
AS
BEGIN
MERGE INTO DW_DATA.dim_employees A
USING ( SELECT employee_id ,first_name_EMPLOYEE , last_name_EMPLOYEE, email_EMPLOYEE, phone_EMPLOYEE, POSITION_NAME_ACTUAL,
              POSITION_DEGREE, SALES_TYPE , HIRE_DATE , salary, chief_id , position_name_previous, position_change_date
FROM DW_CLEANSING.cl_employees) B
ON (a.employee_id = b.employee_id)
WHEN MATCHED THEN
UPDATE SET a.salary = b.salary
WHEN NOT MATCHED THEN
INSERT (a.employee_id ,a.first_name_EMPLOYEE , a.last_name_EMPLOYEE, a.email_EMPLOYEE, a.phone_EMPLOYEE,a.POSITION_NAME_ACTUAL,
a.POSITION_DEGREE, a.SALES_TYPE , a.HIRE_DATE , a.salary, a.chief_id , a.position_name_previous, a.position_change_date,
VALUES (b.employee_id ,b.first_name_EMPLOYEE , b.last_name_EMPLOYEE, b.email_EMPLOYEE, b.phone_EMPLOYEE,b.POSITION_NAME_ACTUAL,
b.POSITION_DEGREE, b.SALES_TYPE , b.HIRE_DATE , b.salary, b.chief_id , b.position_name_previous, b.position_change_date,
COMMIT;
END LOAD_DW_EMPLOYEES;
```

2) To_refcursor

```

31
32 -----to_refcursor-----
33 PROCEDURE LOAD_DW_EMPLOYEES_with_to_refcursor_func
34
35 AS
36 BEGIN
37 DECLARE
38     cursor_id NUMBER (25);
39     cur_count NUMBER (38);
40     query_cur VARCHAR2(2000);
41     TYPE ref_csr IS REF CURSOR;
42     ref_cursor ref_csr;
43     TYPE type_rec IS RECORD
44     (
45         employee_id          NUMBER(5),
46         first_name_EMPLOYEE  VARCHAR2(40),
47         last_name_EMPLOYEE   VARCHAR2(40),
48         email_EMPLOYEE       VARCHAR2(40),
49         phone_EMPLOYEE       VARCHAR2(40),
50         POSITION_NAME_ACTUAL  VARCHAR2(40)

```

Worksheet Query Builder

```

58     one_record type_rec;
59
60 BEGIN
61 query_cur:= 'SELECT employee_id ,first_name_EMPLOYEE , last_name_EMPLOYEE, email_EMPLOYEE, phone_EMPLOYEE,POSITION_NAME_ACTUAL,
62             POSITION_DEGREE, SALES_TYPE , HIRE_DATE , salary, chief_id , position_name_previous, position_change_date
63
64             FROM
65
66             (SELECT stage.employee_id ,stage.first_name_EMPLOYEE , stage.last_name_EMPLOYEE, stage.email_EMPLOYEE,
67             stage.phone_EMPLOYEE, stage.POSITION_NAME_ACTUAL, stage.POSITION_DEGREE, stage.SALES_TYPE ,
68             stage.HIRE_DATE , stage.salary, stage.chief_id , stage.position_name_previous, stage.position_change_date
69
70             FROM DW_CLEANSING.cl_employees source
71             LEFT JOIN dw_data.dim_employees stage
72             ON (source.employee_id=stage.employee_id))';
73
74
75     cursor_id:=DBMS_SQL.open_cursor;
76

```

Worksheet Query Builder

```

75     cursor_id:=DBMS_SQL.open_cursor;
76
77     DBMS_SQL.PARSE(cursor_id, query_cur, DBMS_SQL.NATIVE);
78
79     cur_count:= DBMS_SQL.EXECUTE(cursor_id);
80
81     ref_cursor:= DBMS_SQL.TO_REFCURSOR(cursor_id);
82
83 LOOP
84     FETCH ref_cursor INTO one_record;
85     EXIT WHEN ref_cursor%NOTFOUND;
86 IF (one_record.employee_id IS NULL) THEN
87     INSERT INTO dw_data.dim_employees (employee_id ,first_name_EMPLOYEE , last_name_EMPLOYEE, email_EMPLOYEE,
88     phone_EMPLOYEE,POSITION_NAME_ACTUAL, POSITION_DEGREE, SALES_TYPE , HIRE_DATE , salary,
89     chief_id , position_name_previous, position_change_date)
90     VALUES (SEQ_EMPLOYEES.NEXTVAL,
91     one_record.first_name_EMPLOYEE , one_record.last_name_EMPLOYEE, one_record.email_EMPLOYEE,
92     one_record.phone_EMPLOYEE, one_record.POSITION_NAME_ACTUAL, one_record.POSITION_DEGREE, one_record.SALES_TYPE ,
93     one_record.HIRE_DATE , one_record.salary, one_record.chief_id , one_record.position_name_previous,

```

3) To_cursor_number

```

PROCEDURE LOAD_DW_EMPLOYEES_with_to_cursor_number_func
AS
BEGIN
DECLARE
    l_rc_var1 SYS_REFCURSOR;
    l_n_cursor_id    NUMBER;
    l_n_rowcount     NUMBER;
    l_n_column_count NUMBER;

    l_vc_employee_id    NUMBER(5);
    l_vc_first_name_EMPLOYEE VARCHAR2(40);
    l_vc_last_name_EMPLOYEE VARCHAR2(40);
    l_vc_email_EMPLOYEE  VARCHAR2(40);
    l_vc_phone_EMPLOYEE  VARCHAR2(40);
    l_vc_POSITION_NAME_ACTUAL VARCHAR2(40);
    l_vc_l_vc_POSITION_DEGREE    VARCHAR2(40);
    l_vc_SALES_TYPE             VARCHAR2(40);

```

Worksheet	Query Builder
129	BEGIN
130	OPEN l_rc_var1 FOR
131	'SELECT employee_id , first_name_EMPLOYEE , last_name_EMPLOYEE, email_EMPLOYEE,
132	phone_EMPLOYEE, POSITION_NAME_ACTUAL, POSITION_DEGREE, SALES_TYPE ,
133	HIRE_DATE , salary, chief_id , position_name_previous, position_change_date
134	
135	FROM DW_CLEANSING.cl_employees';
136	
137	l_n_cursor_id:= DBMS_SQL.to_cursor_number(l_rc_var1);
138	dbms_sql.describe_columns(l_n_cursor_id,l_n_column_count,l_ntt_desc_tab);
139	FOR loop_col IN 1..l_n_column_count
140	LOOP
141	dbms_sql.define_column(l_n_cursor_id,loop_col,
142	CASE l_ntt_desc_tab(loop_col).col_name
143	WHEN 'employee_id' THEN
144	l_vc_employee_id
145	
146	WHEN 'first_name_EMPLOYEE' THEN
147	l_vc_first_name_EMPLOYEE

Worksheet	Query Builder
147	l_vc_first_name_EMPLOYEE
148	
149	WHEN 'last_name_EMPLOYEE' THEN
150	l_vc_last_name_EMPLOYEE
151	
152	WHEN 'email_EMPLOYEE' THEN
153	l_vc_email_EMPLOYEE
154	
155	WHEN 'phone_EMPLOYEE' THEN
156	l_vc_last_name_EMPLOYEE
157	
158	WHEN 'POSITION_NAME_ACTUAL' THEN
159	l_vc_POSITION_NAME_ACTUAL
160	
161	WHEN 'POSITION_DEGREE' THEN
162	l_vc_POSITION_DEGREE
163	
164	WHEN 'SALES_TYPE' THEN
165	l_vc_SALES_TYPE
166	
167	WHEN 'HIRE_DATE' THEN
168	l_vc_HIRE_DATE

Worksheet	Query Builder
168	l_vc_HIRE_DATE
169	
170	WHEN 'salary' THEN
171	l_vc_salary
172	
173	WHEN 'chief_id' THEN
174	l_vc_chief_id
175	
176	WHEN 'position_name_previous' THEN
177	l_vc_position_name_previous
178	
179	WHEN 'position_change_date' THEN
180	l_vc_position_change_date
181	
182	END, 50);
183	
184	END LOOP loop_col;
185	LOOP
186	l_n_rowcount:=dbms_sql.fetch_rows(l_n_cursor_id);
187	EXIT
188	WHEN l_n_rowcount=0;
189	FOR loop_col IN 1..l_n_column_count

Worksheet	Query Builder
190	LOOP
191	CASE l_ntt_desc_tab(loop_col).col_name
192	
193	WHEN 'employee_id' THEN
194	dbms_sql.column_value(l_n_cursor_id,loop_col,l_vc_employee_id);
195	
196	WHEN 'first_name_EMPLOYEE' THEN
197	dbms_sql.column_value(l_n_cursor_id,loop_col,l_vc_first_name_EMPLOYEE);
198	
199	WHEN 'last_name_EMPLOYEE' THEN
200	dbms_sql.column_value(l_n_cursor_id,loop_col,l_vc_last_name_EMPLOYEE);
201	
202	WHEN 'email_EMPLOYEE' THEN
203	dbms_sql.column_value(l_n_cursor_id,loop_col,l_vc_email_EMPLOYEE);
204	
205	WHEN 'phone_EMPLOYEE' THEN
206	dbms_sql.column_value(l_n_cursor_id,loop_col,l_vc_phone_EMPLOYEE);
207	
208	WHEN 'POSITION_NAME_ACTUAL' THEN
209	dbms_sql.column_value(l_n_cursor_id,loop_col,l_vc_POSITION_NAME_ACTUAL);
210	

Worksheet	Query Builder
229	WHEN 'position_change_date' THEN
230	dbms_sql.column_value(l_n_cursor_id,loop_col,l_vc_position_change_date);
231	END CASE;
232	END LOOP loop_col;
233	IF(l_vc_employee_id IS NOT NULL) THEN
234	INSERT INTO dw_data.dim_employees(employee_id ,
235	first_name_EMPLOYEE ,
236	last_name_EMPLOYEE,
237	email_EMPLOYEE,
238	phone_EMPLOYEE,
239	POSITION_NAME_ACTUAL,
240	POSITION_DEGREE,
241	SALES_TYPE ,
242	HIRE_DATE ,
243	salary,
244	chief_id ,
245	position_name_previous,
246	position_change_date)
247	
248	VALUES (l_vc_employee_id ,
249	l_vc_first_name_EMPLOYEE ,
250	l_vc_last_name_EMPLOYEE ,

```
Worksheet  Query Builder
250         l_vc_last_name_EMPLOYEE      ,
251         l_vc_email_EMPLOYEE           ,
252         l_vc_phone_EMPLOYEE           ,
253         l_vc_POSITION_NAME_ACTUAL      ,
254         l_vc_l_vc_POSITION_DEGREE     ,
255         l_vc_SALES_TYPE                ,
256         l_vc_HIRE_DATE                 ,
257         l_vc_salary                    ,
258         l_vc_chief_id                  ,
259         l_vc_position_name_previous    ,
260         l_vc_position_change_date);
261     END IF;
262     END LOOP;
263     COMMIT;
264     END;
265
266     END LOAD_DW_EMPLOYEES_with_to_cursor_number_func;
267
268     END pkg_dw_employees_independent;
269
```

Note. All packages rewritten. Let's execute the (and will not forget to truncate tables before executing to make sure packages work correctly)

```
279 TRUNCATE TABLE dim_employees
280
281 exec pkg_dw_employees_independent.LOAD_DW_EMPLOYEES;
```

Script Output x

Task completed in 0.636 seconds

Table DIM_EMPLOYEES truncated.

PL/SQL procedure successfully completed.

```
283 TRUNCATE TABLE dim_employees
284
285 exec pkg_dw_employees_independent.LOAD_DW_EMPLOYEES_with_to_refcursor_func;
```

Script Output x

Task completed in 0.077 seconds

1,000 rows selected.

Table DIM_EMPLOYEES truncated.

PL/SQL procedure successfully completed.

```
287 TRUNCATE TABLE dim_employees
288
289 exec pkg_dw_employees_independent.LOAD_DW_EMPLOYEES_with_to_cursor_number_func;
290 SELECT * FROM dim_employees
```

Script Output x

Task completed in 0.077 seconds

Package Body PKG_DW_EMPLOYEES_INDEPENDENT compiled

3. Business analyses tasks – Reports

3.1. Task 02: CREATE Monthly Reports Layouts

The Main Task is to create Reports Layouts according your Business Solution Proposal, which was developed on Exit Task Module 6.Introduction to DWH.

Required points:

- Refactoring Adhoc SQL, which was developed on Module 7 ETL - Extract, transform and load labwork 02;
- Use Module Clause

Task Results:

Create report layouts:

- Refactoring report layouts
- Put report layouts on Git – Folder BI Tasks – Product Name (author) - Repots

Laboratory Work Summary: At this laboratory work we practised usage of more types of data movement methods, such as:

- Use Execute Immediate with Bind Parameters
- Use DBMS_SQL.TO_REFCURSOR Function
- Use DBMS_SQL.TO_CURSOR_NUMBER Function