Report

Laboratory Work 3

Dmitry Ladutsko

August 11, 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 analysis tasks – Reports

2.1. Task 01: Export Geo Location Reference

<u>The Main Task</u> is to export Geo Location Reference on Denormalized table.

Create Denormalized export table on SB_MBackUp schema.

<u>Required points:</u>

- **Create Denormalized table data using CONNENT_BY**
- **Add Additional Columns to table:**

  Geo_id types: Branch, ROOT, Leaf

  Count of childs of Branch or Root, for Leafs this Field you have fill by NULL

Full path of Dependencies by Example: ROOT -> BRANCH -> BRANCH -> LEAF

```
10  alter session set current_schema=u_dw_references;
11  select
12      *
13  from
14      t_geo_object_links;
15
16  ----------------------------------------------------
17  select * from geo_data_structure;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 38 in 0.01 seconds

	PARENT_GEO_ID	CHILD_GEO_ID	LINK_TYPE_ID
24	247	270	2
25	248	258	2
26	248	264	2
27	248	268	2
28	248	269	2
29	271	272	4
30	271	273	4
31	272	274	5
32	272	275	5
33	272	278	5

SQL Worksheet | History — DmitryOffice

Worksheet | Query Builder

```
16  ----------------------------------------------------
17  select * from geo_data_structure;
18  ----------------------------------------------------
```

Script Output | Query Result | Query Result 1

SQL | Fetched 150 rows in 0.015 seconds

	GEO_ID	GEO_TYPE_ID	GEO_CODE_ID	GEO_TYPE_CODE	GEO_TYPE_DESC	PARENT_GEO_ID	CHILD_GEO_ID	LINK_TYPE_ID
133	386	12	212	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
134	432	12	156	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
135	520	12	830	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
136	420	12	680	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
137	280	52	201	COUNTRY SUB GROUP	Referene: List of All Countries Sub Groups	(null)	(null)	(null)
138	298	12	414	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
139	450	12	170	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
140	342	12	180	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
141	343	12	275	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
142	401	12	188	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
143	422	12	116	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
144	255	11	30	REGIONS	Referene: List of All Continets - Regions	(null)	(null)	(null)
145	261	11	5	REGIONS	Referene: List of All Continets - Regions	(null)	(null)	(null)
146	378	12	882	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
147	434	12	48	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)
148	354	12	360	COUNTRY	Referene: List of All Countries	(null)	(null)	(null)

```
107  alter session set current_schema=u_dw_references;
108  SELECT
109      count(*)
110  FROM
111      geo_denormalized_data
112  WHERE GEO_ID IS NULL;
```

Script Output | Query Result | Query Result 1 | Query R

Task completed in 0.07 seconds

```
Session altered.


  COUNT(*)
----------
         0
```

SQL Worksheet | History

0.48199999 seconds

Worksheet | Query Builder

```
92  LEFT OUTER JOIN lc_geo_systems lgs ON lgs.geo_id = connect_geo_sys_id
93  LEFT OUTER JOIN lc_geo_parts   lgp ON lgp.geo_id = connect_geo_part_id
94  LEFT OUTER JOIN lc_geo_regions lgr ON lgr.geo_id = connect_geo_reg_id
```

Script Output | Query Result

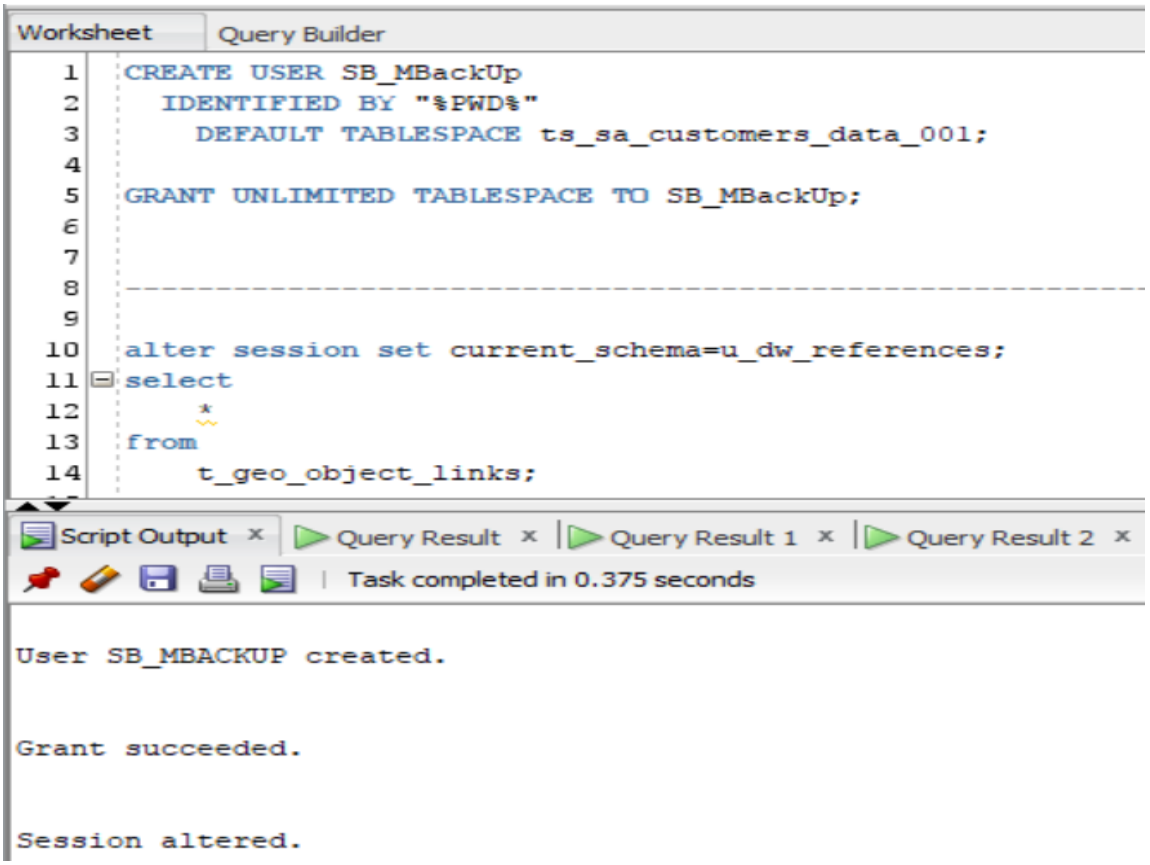
SQL | Fetched 50 rows in 0.021 seconds

	GEO_ID	PARENT_GEO_ID	CHILD_AMOUNT	LVL	ID_TYPE	PATH
1	242	(null)	6	1	ROOT	/242
2	243	242	3	2	BRANCH(1)	/242/243
3	261	243	(null)	3	BRANCH(2)	/242/243/261
4	266	243	(null)	3	BRANCH(2)	/242/243/266
5	267	243	(null)	3	BRANCH(2)	/242/243/267
6	244	242	5	2	BRANCH(1)	/242/244
7	250	244	(null)	3	BRANCH(2)	/242/244/250
8	251	244	(null)	3	BRANCH(2)	/242/244/251
9	254	244	(null)	3	BRANCH(2)	/242/244/254
10	259	244	(null)	3	BRANCH(2)	/242/244/259
11	262	244	(null)	3	BRANCH(2)	/242/244/262
12	245	242	1	2	BRANCH(1)	/242/245
13	260	245	(null)	3	BRANCH(2)	/242/245/260
14	246	242	5	2	BRANCH(1)	/242/246
15	252	246	(null)	3	BRANCH(2)	/242/246/252
16	253	246	(null)	3	BRANCH(2)	/242/246/253

<u>Task Results:</u>

Create required objects:

- **Create New Schema SB_MBackUp and New Default TableSpace**
- **Put objects script to Git.**



- **Prepare Document with Screenshot of Data on Denormalized table**
- **Prepare load script and put it to GIT**

2.2. Task 02: Analyze Business hierarchy Reference Analyses

<u>The Main Task</u> is to create hierarch analyses of any Dimension, according yours Solution Proposal and DWH Solution Concept from Module 6. Introduction to DWH

<u>Required points:</u>

- **Create Denormalized table data using CONNENT_BY**
- **Use START WITH Clause**
- **Use CONNECT_BY_ROOT to analyses any Branch levels**
- **Analyze Main Root Branch, and 2 Sub Branches**

```
CREATE TABLE SA_EMPLOYEES (
    first_name          VARCHAR2(40) NOT NULL,
    last_name           VARCHAR2(40) NOT NULL,
    email               VARCHAR2(40) NOT NULL,
    phone               VARCHAR2(40) NOT NULL,
    POSITION_NAME       VARCHAR2(40) NOT NULL,
    POSITION_GRADE      VARCHAR2(40) NOT NULL,
    HIRE_DATE           DATE NOT NULL
);
```

```
CREATE TABLE SA_EMPLOYEES (
    employee_id             NUMBER(5) not null,
    first_name_EMPLOYEE     VARCHAR2(40) NOT NULL,
    last_name_EMPLOYEE      VARCHAR2(40) NOT NULL,
    email_EMPLOYEE          VARCHAR2(40) NOT NULL,
    phone_EMPLOYEE          VARCHAR2(40) NOT NULL,
    POSITION_NAME_ACTUAL    VARCHAR2(40) NOT NULL,
    POSITION_DEGREE         VARCHAR2(40) NOT NULL,
    SALES_TYPE              VARCHAR2(40) NOT NULL,
    HIRE_DATE               DATE NOT NULL,
    salary                  int not null,
    chief_id                int not null,
    position_name_previous  VARCHAR2(40) NOT NULL,
    position_change_date    DATE not null
);
```

Here, as you san see, I added more data into sa_employees table (Left - previous version, right - new)

```
alter session set current_schema = SA_PRODUCTS;
select
    *
from
    sa_employees
CONNECT BY PRIOR
    employee_id = chief_id
START WITH
    chief_id = 0;
```

Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 6

SQL | Fetched 150 rows in 0.141 seconds

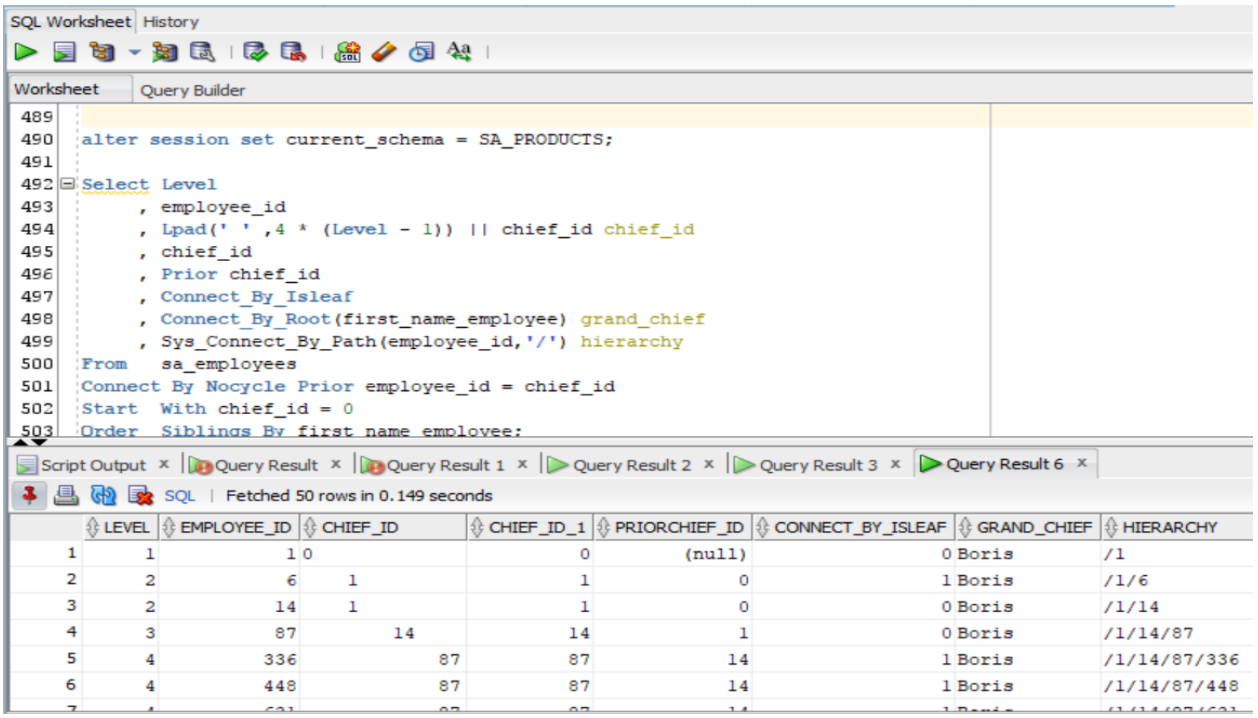
PLOYEE	POSITION_NAME_ACTUAL	POSITION_DEGREE	SALES_TYPE	HIRE_DATE	SALARY	CHIEF_ID	POSITION_NAME_PREVIOUS	POSITION_CHANGE_DATE
	Product-line sales manager	Senior	B2B	07-JUL-21	8699	0	Product-line sales manager	24-AUG-21
	Sales Manager	Middle	B2B	29-MAR-21	4842	1	Manager	15-NOV-21
	Manager	Junior	Enterprice	08-MAY-21	4589	2	regional sales manager	15-SEP-21
	Manager	Top	B2C	28-AUG-21	6158	71	regional sales manager	21-NOV-21
	Product-line sales manager	Middle	Direct	24-JUN-21	3101	71	Sales Manager	02-DEC-21
	Manager	Middle	Enterprice	17-MAY-20	6477	71	Product-line sales manager	23-JUN-20

```
alter session set current_schema = SA_PRODUCTS;
select
    *
from
    sa_employees
CONNECT BY PRIOR
    employee_id = chief_id
START WITH
    chief_id = 0;
```

Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 6

SQL | Fetched 150 rows in 0.141 seconds

EMPLOYEE_ID	FIRST_NAME_EMPLOYEE	LAST_NAME_EMPLOYEE	EMAIL_EMPLOYEE	PHONE_EMPLOYEE	POSITION_NAME_ACTUAL	POSITION_DEGREE	
1	Boris	Brown	nagibator@apple.com	5226460	Product-line sales manager	Senior	B
2	Miron	Johnaton	genius@apple.com	7190334	Sales Manager	Middle	B
71	Abrigel	Brown	gooof@apple.com	7439171	Manager	Junior	E
175	John	Alumos	helloljdnjn267@apple.com	8284268	Manager	Top	B
267	Vika	Grey	dffsfsmd@apple.com	4652698	Product-line sales manager	Middle	D
421	Artem	Smith	genesis12@apple.com	8546011	Manager	Middle	E



In the pictures above you can see how **employees** related with **chiefs**. And who is a chief for every employee, who is a **grand chief** and their 4 - staged **hierarchy**.

<u>Task Results:</u>

Create required objects:

- **Prepare Document with Screenshot of analyses Data result**
- **Prepare script and put it to GIT**

Note. All scripts stored on GitHub.

Laboratory work summary:

At this lab we have learned how we can carry out a hierarchy schema using different statements, such as :

- Connect_by
- Connect_by_root

Specified with such clauses as **Start With** and **Level** pseudocolumn which shows parent and child rows. **Now we have** much more understanding about hierarchy usage.