Lab Report #3
Sadovskaya Veronika

Task 01 – Install and expand load of external references T_Languages

Step 1: Default DataBase data files location.

```
--Change parameter:Default DataBase data files location
ALTER SYSTEM SET db_create_file_dest = '/oracle/u02/oradata/VSadovskaiadb';
```

Script Output | Task completed in 0,025 seconds

```
System SET altered.
```

Figure 1.1

Step 2: create tablespaces.

```
TABLESPACE TS_DW_DATA_01 created.

TABLESPACE TS_DW_IDX_01 created.

TABLESPACE TS_PERSONS_DATA_01 created.

TABLESPACE TS_PERSONS_IDX_01 created.

TABLESPACE TS_REFERENCES_DATA_01 created.

TABLESPACE TS_REFERENCES_EXT_DATA_01 created.

TABLESPACE TS_REFERENCES_IDX_01 created.
```

Figure 1.2

Step 3: create users.

```
    IDENTIFIED BY "&PWD&";
```

Script Output | Task completed in 0,118 seconds

```
User U_DW_DATA created.

Grant succeeded.

User U_DW_EXT_REFERENCES created.

Grant succeeded.

User U_DW_PERSONS created.

Grant succeeded.

User U_DW_REFERENCES created.
```

Figure 1.3

Step 4: create directory.

```
CREATE OR REPLACE DIRECTORY ext_references
AS
  '/mnt/ext_references/';

GRANT READ ON DIRECTORY ext_references TO u_dw_ext_references;
GRANT WRITE ON DIRECTORY ext_references TO u_dw_ext_references;
```

Script Output

Task completed in 0,084 seconds

```
Directory EXT_REFERENCES created.


Grant succeeded.


Grant succeeded.
```

Figure 1.4

Step 5: run the scripts the following folders in certain order:

a) /u_dw_references
b) /u_dw_ext_references
c) /u_dw_common

Step 6: showing result:

a) t_localizations

```
SELECT * FROM u_dw_references.t_localizations;
```

Query Result

SQL | All Rows Fetched: 5 in 0,063 seconds

	LOCALIZATION_ID	LOCALIZATION_CODE	LOCALIZATION_DESC	LOCALIZATION_DESC_ENS	LNG_ID	CONTRY_ID	IS_DEFAULT
1	-1	n.a.	Not Available	Not Available	(null)	(null)	(null)
2	-2	n.d.	Not Defined	Not Defined	(null)	(null)	(null)
3	1	en-US	English	English	(null)	(null)	1
4	2	ru-RU	Russian	Русский	(null)	(null)	(null)
5	3	be-BY	Belarussian	Беларускі	(null)	(null)	(null)

Figure 1.5

b) cu_languages

```
18  select * from U_DW_REFERENCES.cu_languages;
```

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

SQL | Fetched 50 rows in 0,009 seconds

	LNG_ID	LNG_3C_CODE	LNG_2B_CODE	LNG_2T_CODE	LNG_1C_CODE	LNG_SCOPE_ID	LNG_TYPE_ID	LNG_DESC
1	2212	aaa	(null)	(null)	(null)	1	5	Ghotuo
2	2213	aab	(null)	(null)	(null)	1	5	Alumu-Tesu
3	2214	aac	(null)	(null)	(null)	1	5	Ari
4	2215	aad	(null)	(null)	(null)	1	5	Amal
5	2216	aae	(null)	(null)	(null)	1	5	Arbëreshë Albanian
6	2217	aaf	(null)	(null)	(null)	1	5	Aranadan
7	2218	aag	(null)	(null)	(null)	1	5	Ambrak
8	2219	aah	(null)	(null)	(null)	1	5	Abu' Arapesh
9	2220	aai	(null)	(null)	(null)	1	5	Arifama-Miniafia
10	2221	aak	(null)	(null)	(null)	1	5	Ankave
11	2222	aal	(null)	(null)	(null)	1	5	Afade
12	2223	aam	(null)	(null)	(null)	1	5	Aramanik

Figure 1.6

c) w_lng_links

```
25  SELECT * FROM u_dw_references.w_lng_links;
```

Query Result

SQL | All Rows Fetched: 0 in 0,012 seconds

PARENT_...	CHILD_LN...	LINK_TYP...

Figure 1.7

d) cu_lng_scopes

```
16  select * from U_DW_REFERENCES.cu_lng_scopes;
```

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

SQL | All Rows Fetched: 3 in 0,055 seconds

	LNG_SCOPE_ID	SRC_SCOPE_CODE	LNG_SCOPE_CODE	LNG_SCOPE_DESC	LOCALIZATION_ID
1	1	I	I	Individual	1
2	2	M	M	Macrolanguage	1
3	3	S	S	Special	1

Figure 1.8

e) cu_lng_types

```
17  select * from U_DW_REFERENCES.cu_lng_types;
```

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

SQL | All Rows Fetched: 6 in 0,013 seconds

	LNG_TYPE_ID	SRC_TYPE_CODE	LNG_TYPE_CODE	LNG_TYPE_DESC	LOCALIZATION_ID
1	1	A	A	Ancient	1
2	2	C	C	Constructed	1
3	3	E	E	Extinct	1
4	4	H	H	Historical	1
5	5	L	L	Living	1
6	6	S	S	Special	1

Figure 1.9

Step 7: create script to show all created tables and views.

```
34  SELECT Table_Name, OWNER
35  FROM SYS.ALL_TABLES
36  WHERE OWNER IN ('U_DW_REFERENCES', 'U_DW_EXT_REFERENCES', 'U_DW_COMMON')
37  UNION
38  SELECT View_Name, OWNER
39  FROM SYS.ALL_VIEWS
40  WHERE OWNER IN ('U_DW_REFERENCES', 'U_DW_EXT_REFERENCES', 'U_DW_COMMON')
41  ORDER BY 1;
```

Query Result

SQL | All Rows Fetched: 21 in 0,583 seconds

	TABLE_NAME	OWNER
1	CLS_LANGUAGES_ISO693	U_DW_EXT_REFERENCES
2	CLS_LNG_MACRO2IND_ISO693	U_DW_EXT_REFERENCES
3	CU_LANGUAGES	U_DW_REFERENCES
4	CU_LNG_SCOPES	U_DW_REFERENCES
5	CU_LNG_TYPES	U_DW_REFERENCES
6	LC_LNG_SCOPES	U_DW_REFERENCES
7	LC_LNG_TYPES	U_DW_REFERENCES
8	T_EXT_LANGUAGES_ISO693	U_DW_EXT_REFERENCES
9	T_EXT_LNG_MACRO2IND_ISO693	U_DW_EXT_REFERENCES
10	T_LANGUAGES	U_DW_REFERENCES
11	T_LNG_LINKS	U_DW_REFERENCES
12	T_LNG_SCOPES	U_DW_REFERENCES

Figure 1.10 – Part 1

```
34  SELECT Table_Name, OWNER
35  FROM SYS.ALL_TABLES
36  WHERE OWNER IN ('U_DW_REFERENCES', 'U_DW_EXT_REFERENCES', 'U_DW_COMMON')
37  UNION
38  SELECT View_Name, OWNER
39  FROM SYS.ALL_VIEWS
40  WHERE OWNER IN ('U_DW_REFERENCES', 'U_DW_EXT_REFERENCES', 'U_DW_COMMON')
41  ORDER BY 1;
```

Query Result ×

SQL | All Rows Fetched: 21 in 0,583 seconds

	TABLE_NAME	OWNER
10	T_LANGUAGES	U_DW_REFERENCES
11	T_LNG_LINKS	U_DW_REFERENCES
12	T_LNG_SCOPES	U_DW_REFERENCES
13	T_LNG_TYPES	U_DW_REFERENCES
14	T_LOCALIZATIONS	U_DW_REFERENCES
15	VL_LNG_SCOPES	U_DW_REFERENCES
16	VL_LNG_TYPES	U_DW_REFERENCES
17	W_LANGUAGES	U_DW_REFERENCES
18	W_LNG_LINKS	U_DW_REFERENCES
19	W_LNG_SCOPES	U_DW_REFERENCES
20	W_LNG_TYPES	U_DW_REFERENCES
21	W_LOCALIZATIONS	U_DW_REFERENCES

Figure 1.11 – Part 2

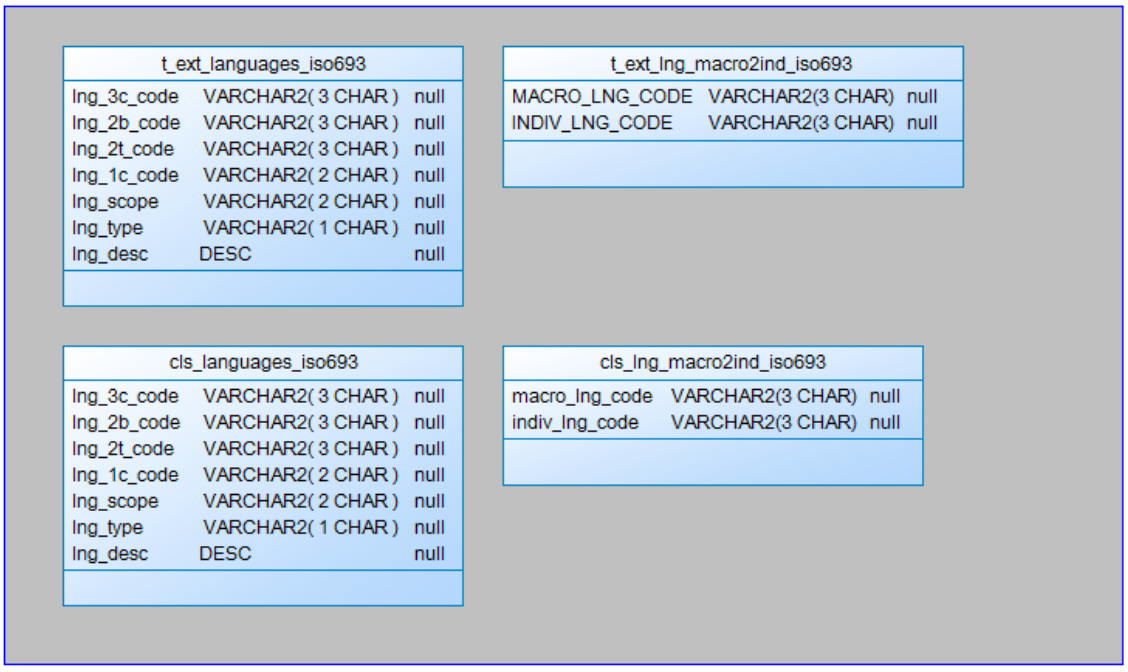

Figure 1.12 – External references tables

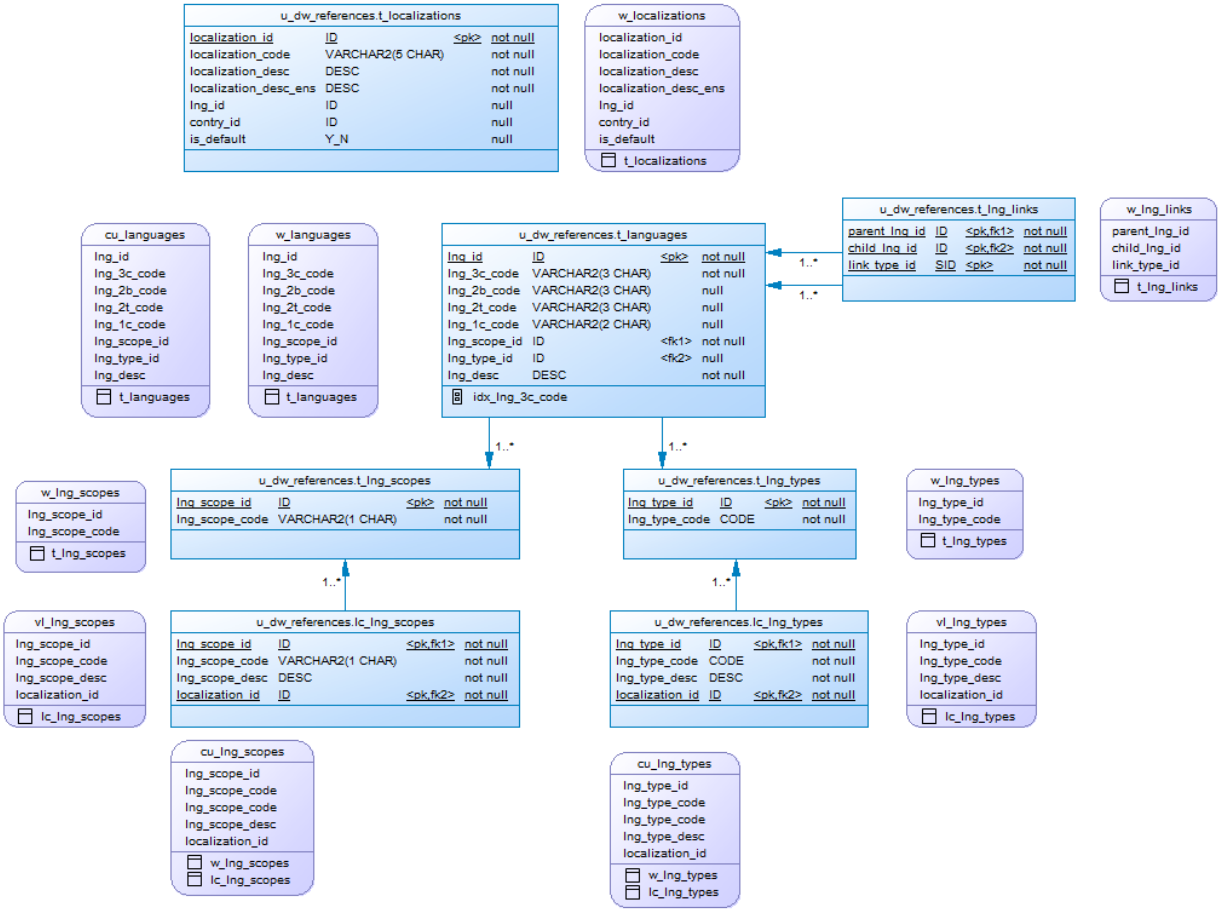

Figure 1.13 – The Physical Diagram of t_languages

Step 8: create DataFlow: Sketch Diagram of loading external References.

Source		Procedure		Object
t_ext_languages_iso693	->	load_cls_languages	->	cls_languages_iso693
cls_languages_iso693	->	load_ref_lng_scopes	->	w_lng_scopes
w_lng_scopes	->	load_ref_lng_scopes	->	vl_lng_scopes
cls_languages_iso693	->	load_ref_lng_types	->	w_lng_types
w_lng_types	->	load_ref_lng_types	->	vl_lng_types lng
cls_languages_iso693 w_lng_scopes w_lng_types	->	load_ref_lanuages	->	w_languages
t_ext_lng_macro2ind_iso693	->	load_cls_links_macro2indiv	->	cls_lng_macro2ind_iso693
cls_lng_macro2ind_iso693 w_languages	->	load_ref_lng_links_macro	->	w_lng_links

Figure 1.14

Task 02 – Create load process for External references T_Countries

Step 1: create SQL scripts to show All created Tables and Views.

```
SELECT Table_Name, OWNER
FROM SYS.ALL_TABLES
WHERE OWNER IN ('U_DW_REFERENCES', 'U_DW_EXT_REFERENCES', 'U_DW_COMMON')
UNION
SELECT View_Name, OWNER
FROM SYS.ALL_VIEWS
WHERE OWNER IN ('U_DW_REFERENCES', 'U_DW_EXT_REFERENCES', 'U_DW_COMMON')
ORDER BY 1;
```

Query Result | SQL | All Rows Fetched: 75 in 0,719 seconds

	TABLE_NAME	OWNER
10	CU_CNTR_GROUP_SYSTEMS	U_DW_REFERENCES
11	CU_CNTR_SUB_GROUPS	U_DW_REFERENCES
12	CU_COUNTRIES	U_DW_REFERENCES
13	CU_GEO_PARTS	U_DW_REFERENCES
14	CU_GEO_REGIONS	U_DW_REFERENCES
15	CU_GEO_SYSTEMS	U_DW_REFERENCES
16	CU_LANGUAGES	U_DW_REFERENCES
17	CU_LNG_SCOPES	U_DW_REFERENCES
18	CU_LNG_TYPES	U_DW_REFERENCES
19	LC_CNTR_GROUPS	U_DW_REFERENCES
20	LC_CNTR_GROUP_SYSTEMS	U_DW_REFERENCES
21	LC_CNTR_SUB_GROUPS	U_DW_REFERENCES
22	LC_COUNTRIES	U_DW_REFERENCES

Figure 2.1

Step 2: create SQL: Showing result of data on main objects:

a) t_geo_systems

```
SELECT * FROM u_dw_references.t_geo_systems;
```

Query Result | Query Result 1 | Query Result 2 | SQL | All Rows Fetched: 1 in 0,029 seconds

	GEO_ID	GEO_SYSTEM_ID
1	483	1

Figure 2.2

b) t_cntr_group_systems

```
SELECT * FROM u_dw_references.t_cntr_group_systems;
```

Query Result | Query Result 1 | Query Result 2 | Query Re

SQL | All Rows Fetched: 1 in 0,052 seconds

	GEO_ID	GRP_SYST...
1	512	1

Figure 2.3

c) t_cntr_groups

```
SELECT * FROM u_dw_references.t_cntr_groups;
```

Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 2 in 0,029 seconds

	GEO_ID	GROUP_ID
1	513	2
2	514	3

Figure 2.3

d) t_cntr_sub_groups

```
SELECT * FROM u_dw_references.t_cntr_sub_groups;
```

Query Result | Query Result 1 | Query Result 2 | Query

SQL | All Rows Fetched: 8 in 0,052 seconds

	GEO_ID	SUB_GROUP_ID
1	515	204
2	516	203
3	517	302
4	518	303
5	519	202
6	520	301
7	521	201
8	522	205

Figure 2.4

e) t_countries

```
SELECT * FROM u_dw_references.t_countries;
```

Query Result | Query Result 1 | Query Result 2

SQL | Fetched 50 rows in 0,064 seconds

	GEO_ID	COUNTRY_ID
1	242	328
2	243	512
3	244	535
4	245	578
5	246	124
6	247	534
7	248	706
8	249	226
9	250	762

Figure 2.5

f) t_geo_parts

```
SELECT * FROM u_dw_references.t_geo_parts;
```

Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 6 in 0,043 seconds

	GEO_ID	PART_ID
1	484	419
2	485	2
3	486	21
4	487	142
5	488	9
6	489	150

Figure 2.6

g) t_geo_regions

```
SELECT * FROM u_dw_references.t_geo_regions;
```

Query Result | Query Result 1 | Query Result 2 | Q

SQL | All Rows Fetched: 22 in 0,032 seconds

	GEO_ID	REGION_ID
1	490	57
2	491	14
3	492	15
4	493	143
5	494	34
6	495	11
7	496	30
8	497	54
9	498	145
10	499	155

Figure 2.7

h) cu_geo_systems

```
SELECT * FROM u_dw_references.cu_geo_systems;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4 | Query R

SQL | All Rows Fetched: 1 in 0,036 seconds

	GEO_ID	SRC_GEO_SYSTEM_ID	GEO_SYSTEM_CODE	GEO_SYSTEM_DESC	LOCALIZATION_ID
1	483	1	WORLD	The UN World structure	1

Figure 2.8

i) cu_cntr_group_systems

```
SELECT * FROM u_dw_references.cu_cntr_group_systems;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4 | Query Result 5 | Qu

SQL | All Rows Fetched: 1 in 0,028 seconds

	GEO_ID	GRP_SYSTEM_ID	GRP_SYSTEM_CODE	GRP_SYSTEM_DESC	LOCALIZATION_ID
1	512	1	MAIN	Selected Economic and other groupings	1

Figure 2.9

j) cu_cntr_groups

```
SELECT * FROM u_dw_references.cu_cntr_groups;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query R

SQL | All Rows Fetched: 2 in 0,028 seconds

	GEO_ID	GROUP_ID	GROUP_CODE	GROUP_DESC	LOCALIZATION_ID
1	513	2	-	Economic groupings	1
2	514	3	-	Unions groupings	1

Figure 2.10

k) cu_cntr_sub_groups

```
SELECT * FROM u_dw_references.cu_cntr_sub_groups;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4 | Query Result 5 | Query R

SQL | All Rows Fetched: 8 in 0,035 seconds

	GEO_ID	SUB_GROUP_ID	SUB_GROUP_CODE	SUB_GROUP_DESC	LOCALIZATION_ID
1	516	203	-	Small island developing states	1
2	515	204	-	Transition countries	1
3	519	202	-	Landlocked developing countries	1
4	522	205	-	Developed countries	1
5	521	201	-	Least developed countries	1
6	520	301	-	Commonwealth of Independent States	1
7	517	302	-	Transition countries of South-Eastern Europe	1
8	518	303	-	European Union	1

Figure 2.11

l) cu_countries

```
SELECT * FROM u_dw_references.cu_countries;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4 | Query Result 5 | Query Result 6

SQL | Fetched 50 rows in 0,051 seconds

	GEO_ID	COUNTRY_ID	COUNTRY_CODE_A2	COUNTRY_CODE_A3	REGION_DESC	LOCALIZATION_ID
1	413	152	CL	CHL	Chile	1
2	329	450	MG	MDG	Madagascar	1
3	480	830	-	-	Channel Islands	1
4	342	408	-	PRK	Democratic People's Republic of Korea	1
5	244	535	BQ	BES	Bonaire, Saint Eustatius and Saba	1
6	361	188	CR	CRI	Costa Rica	1
7	297	768	TG	TGO	Togo	1
8	392	156	CN	CHN	China	1
9	330	203	CZ	CZE	Czech Republic	1
10	256	232	ER	ERI	Eritrea	1
11	389	364	IR	IRN	Iran (Islamic Republic of)	1

Figure 2.12

m) cu_geo_parts

```
SELECT * FROM u_dw_references.cu_geo_parts;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | All Rows Fetched: 6 in 0,036 seconds

	GEO_ID	PART_ID	PART_CODE	PART_DESC	LOCALIZATION_ID
1	488	9	-	Oceania	1
2	486	21	-	Northern America	1
3	485	2	-	Africa	1
4	489	150	-	Europe	1
5	487	142	-	Asia	1
6	484	419	-	Latin America and the Caribbean	1

Figure 2.13

n) cu_geo_regions

```
SELECT * FROM u_dw_references.cu_geo_regions;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | All Rows Fetched: 22 in 0,043 seconds

	GEO_ID	SRC_CONTINENT_ID	REGION_CODE	REGION_DESC	LOCALIZATION_ID
1	491	14	-	Eastern Africa	1
2	503	18	-	Southern Africa	1
3	505	151	-	Eastern Europe	1
4	497	54	-	Melanesia	1
5	495	11	-	Western Africa	1
6	502	5	-	South America	1
7	501	21	-	Northern America	1
8	494	34	-	Southern Asia	1

Figure 2.14

Step 3: create DataFlow: Sketch Diagram of loading external References

Source	->	Procedure	->	Object
t_ext_geo_countries_iso3166	->	load_cls_languages_alpha3	->	cls_geo_countries_iso3166
t_ext_geo_countries2_iso3166	->	load_cls_languages_alpha2	->	cls_geo_countries2_iso3166
cls_geo_countries_iso3166	->	load_ref_geo_countries	->	w_countries
w_countries cls_geo_countries_iso3166 cls_geo_countries2_iso3166	->	load_ref_geo_countries	->	vl_countries
t_ext_geo_structure_iso3166	->	load_cls_geo_structure	->	cls_geo_structure_iso3166
t_ext_cntr2structure_iso3166	->	load_cls_geo_structure2cntr	->	cls_cntr2structure_iso3166
cls_geo_structure_iso3166	->	load_ref_geo_systems	->	w_geo_systems
w_geo_systems cls_geo_structure_iso3166	->	load_ref_geo_systems	->	vl_geo_systems
cls_geo_structure_iso3166	->	load_ref_geo_parts	->	w_geo_parts
w_geo_parts cls_geo_structure_iso3166	->	load_ref_geo_parts	->	vl_geo_parts
cls_geo_structure_iso3166	->	load_ref_geo_regions	->	w_geo_regions
w_geo_regions cls_geo_structure_iso3166	->	load_ref_geo_regions	->	vl_geo_regions
t_ext_cntr_grouping_iso3166	->	load_cls_countries_grouping	->	cls_cntr_grouping_iso3166
t_ext_cntr2grouping_iso3166	->	load_cls_countries2groups	->	cls_cntr2grouping_iso3166
cls_cntr_grouping_iso3166	->	load_ref_cntr_group_systems	->	w_cntr_group_systems
w_cntr_group_systems cls_cntr_grouping_iso3166	->	load_ref_cntr_group_systems	->	vl_cntr_group_systems
cls_cntr_grouping_iso3166	->	load_ref_cntr_groups	->	w_cntr_groups
w_cntr_groups cls_cntr_grouping_iso3166	->	load_ref_cntr_groups	->	vl_cntr_groups
cls_cntr_grouping_iso3166	->	load_ref_cntr_sub_groups	->	w_cntr_sub_groups
w_cntr_sub_groups cls_cntr_grouping_iso3166	->	load_ref_cntr_sub_groups	->	vl_cntr_sub_groups
cls_geo_structure_iso3166 w_geo_objects	->	load_lnk_geo_structure	->	w_geo_object_links
cls_cntr2structure_iso3166 w_countries w_geo_regions	->	load_lnk_geo_countries	->	w_geo_object_links
cls_cntr_grouping_iso3166 w_geo_objects	->	load_lnk_cntr_grouping	->	w_geo_object_links
cls_cntr2grouping_iso3166 w_cntr_sub_groups w_countries	->	load_lnk_cntr2groups	->	w_geo_object_links

Figure 2.15

Step 4: prepare The Physical Diagram of T_Countries

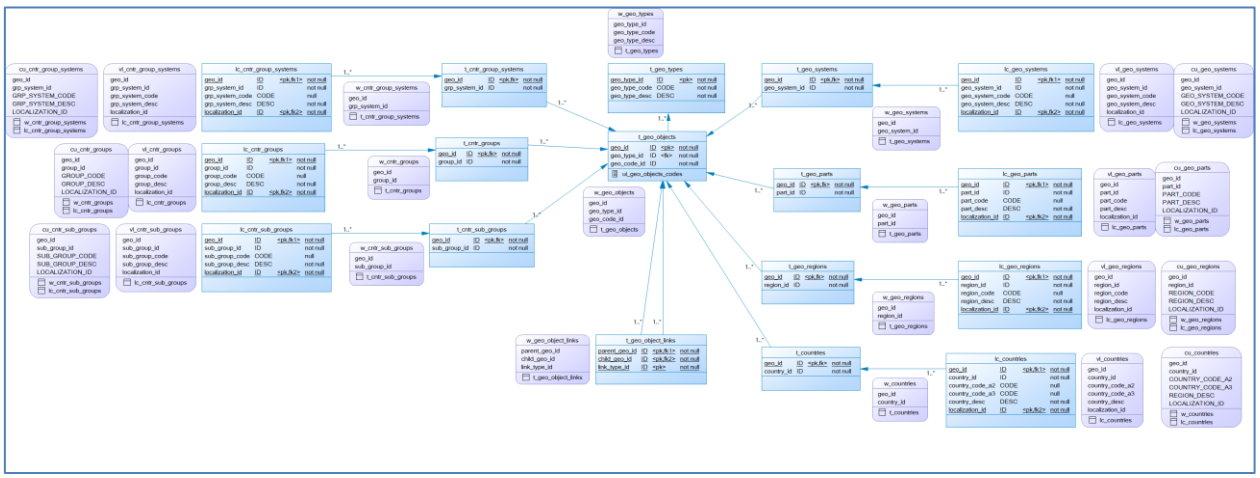

Figure 2.16

Task 03 – Solution concept – Business background

As an example of a business idea, consider the work of the Belarusian chain of restaurants of Georgian cuisine.

Brief overview of the chain of restaurants Khinkalnya:

Khinkalnya is a family, they cherish the recipes of their favorite dishes and cook only the most delicious: khinkali with fragrant broth and juicy filling, khachapuri boats, warming kharcho and tender chikhirtma, spicy kebab and Georgian pkhali. And the most delicious homemade Napoleon. In Khinkalne, as at home, you can celebrate the holiday with sparkling Bedagoni and enjoy an ordinary day with a glass of homemade Kakhetian wine. Gather the whole family at a large table or make an appointment with a dear person, or just drop in for a delicious lunch. Everyone is comfortable in Khinkaln: there is a children's menu and comfortable chairs, as well as entertainment for little guests.

Let this chain of restaurants face certain difficulties related to incorrect data handling. Some of the possible problems:

1. lack of business intelligence from multiple sources;
2. reduced query and system performance;
3. lack of timely access to data;
4. the lack of historical intelligence.

We offer the customer to use DWH(we will consider 2 types of storage schemes: star scheme and snowflake scheme) for the following reasons:

1. Access to the required data. Since the chain is quite large, you need to collect permissions and accesses to receive data from different sources. Each department in such a situation, as a rule, has its own databases with its own passwords, which will need to be requested separately. At DWH, everything you need will already be at hand in finished form. You can just go and pull the necessary statistics there.

2. Keeping the right data. Data in DWH is not lost and is stored in a form convenient for decision making: there are historical records, there are aggregated values. The operational database may not have this information.

3. Sustainability of business systems. DWH is optimized for the work of analysts, and these guys can request very large amounts of information. If they do this with the help of DWH, it's okay, even if their request will be processed for a very long time. And if you request too many records from a regular server database, it can go into failure before the end of the request from analytics and create problems for other systems. DWH eliminates the risk of analysts hanging or breaking something.

Business requirements:

1. chain has several restaurants in different countries. Each restaurant has its own geographical location. The warehouse must take this fact into account;

2. statistics of increase or decrease in profits during promotions;

3. statistics of ordered dishes for different periods of time (during the day, during the week, during the season of the year);

4. calculation of information about visits to restaurants in different regions, cities or even districts of the same city;

5. calculation of statistics of ordered dishes through delivery monthly (you can collect information for large time periods due to the time hierarchy)

Technical requirements:

1. keep info from the beginning of business;
2. persistent and observable storage access;
3. ability to process large amounts of information per day (100 thousands rows);
4. high performance and high availability;
5. all the information must be protected.

Task 04 – Develop Star-Scheme physical diagram

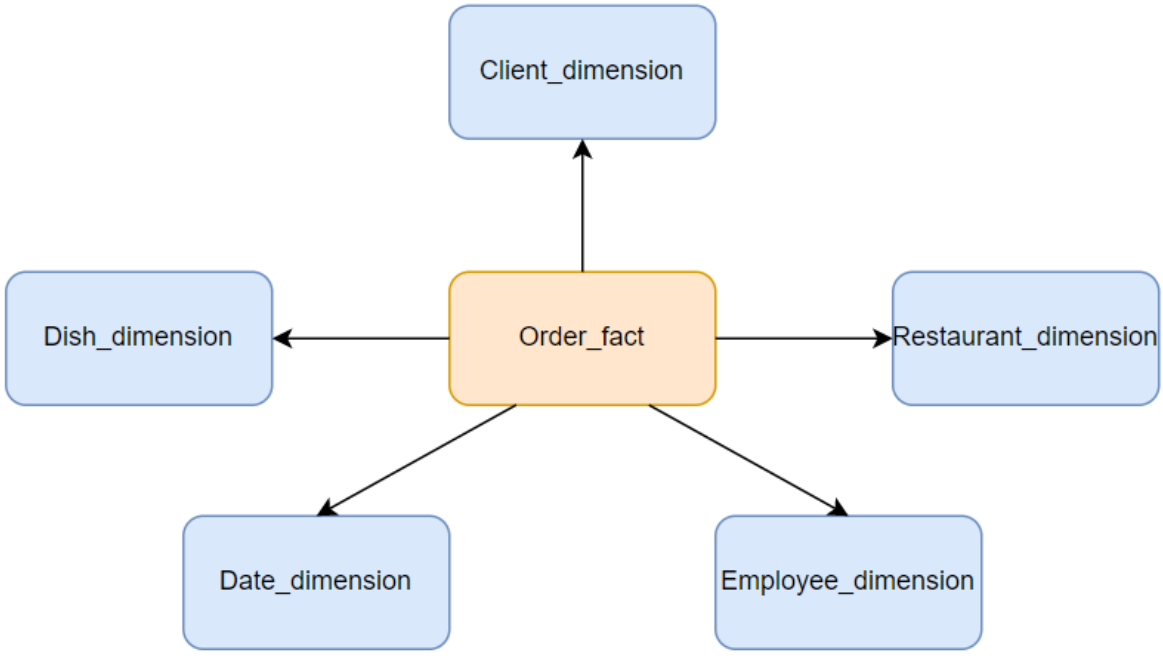

Figure 4.1 – Star-Scheme logical diagram

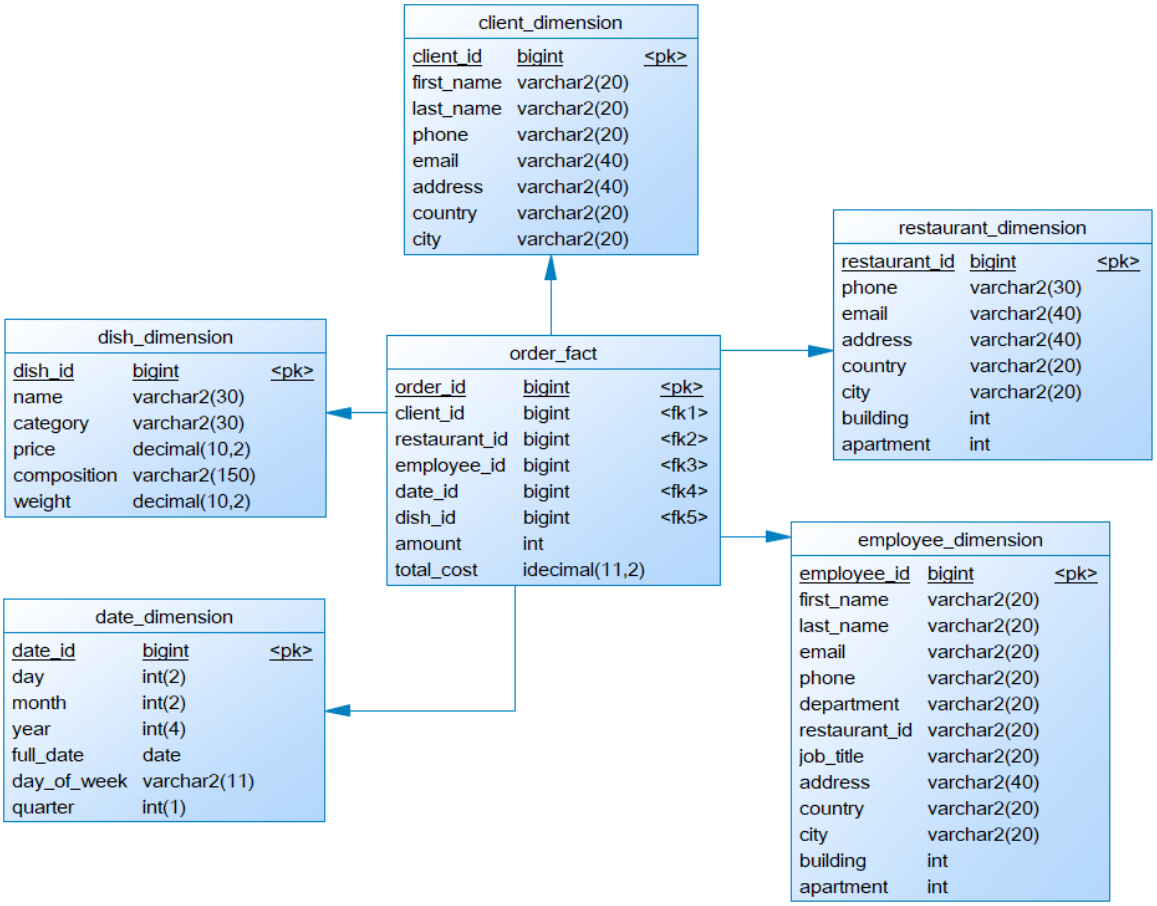

Figure 4.2 – Star-Scheme physical diagram

Task 05 – Develop SnowFlake physical diagram

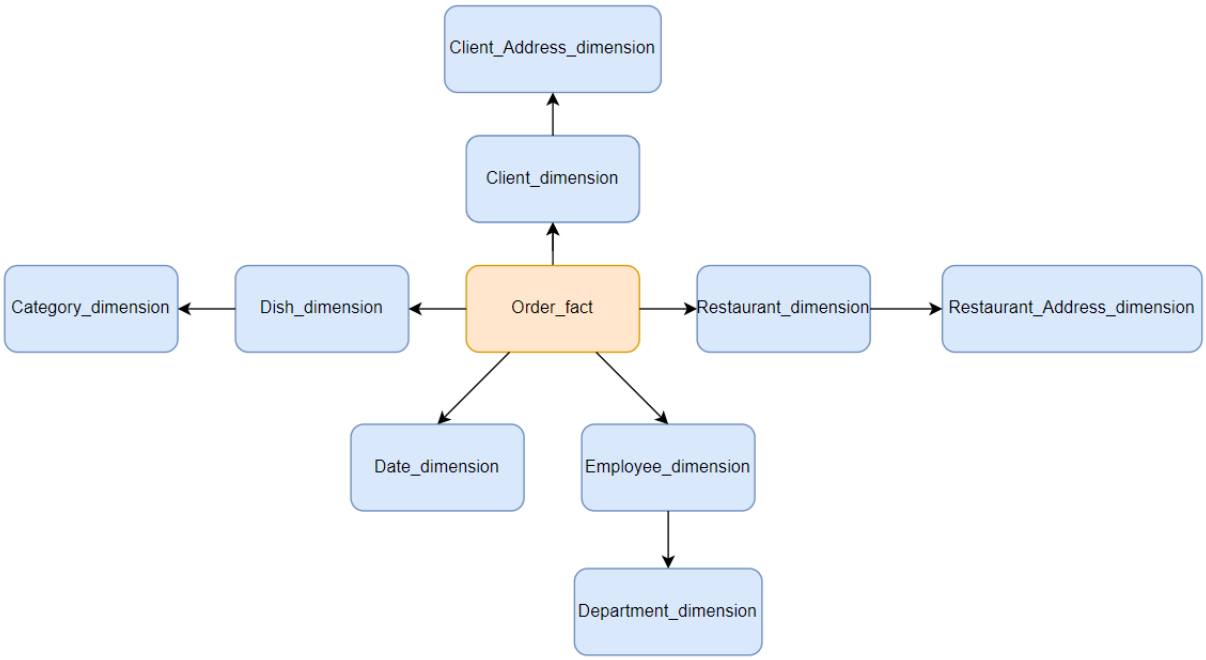

Figure 5.1 – SnowFlake-Scheme logical diagram

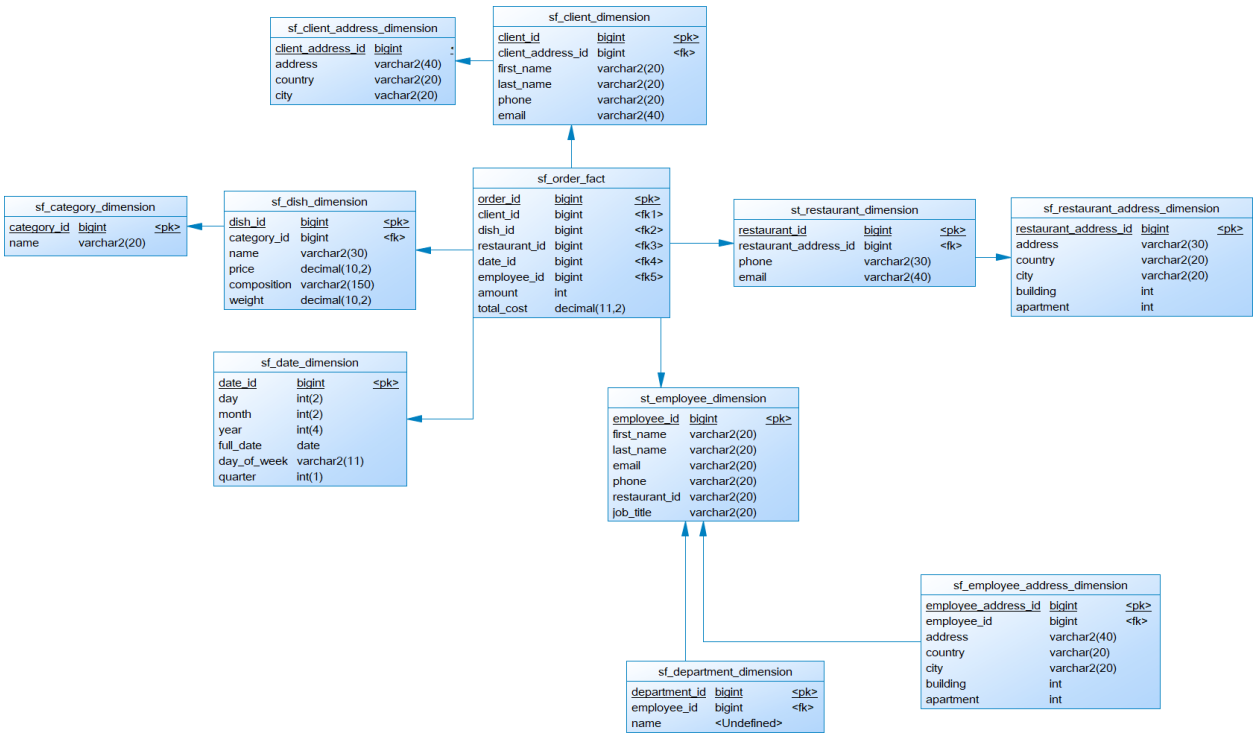
sf_client_address_dimension
client_address_id bigint
address varchar2(40)
country varchar2(20)
city vachar2(20)
sf_client_dimension
client_id bigint <pk>
client_address_id bigint <fk>
first_name varchar2(20)
last_name varchar2(20)
phone varchar2(20)
email varchar2(40)
sf_category_dimension
category_id bigint <pk>
name varchar2(20)
sf_dish_dimension
dish_id bigint <pk>
category_id bigint <fk>
name varchar2(30)
price decimal(10,2)
composition varchar2(150)
weight decimal(10,2)
sf_order_fact
order_id bigint <pk>
client_id bigint <fk1>
dish_id bigint <fk2>
restaurant_id bigint <fk3>
date_id bigint <fk4>
employee_id bigint <fk5>
amount int
total_cost decimal(11,2)
st_restaurant_dimension
restaurant_id bigint <pk>
restaurant_address_id bigint <fk>
phone varchar2(30)
email varchar2(40)
sf_restaurant_address_dimension
restaurant_address_id bigint <pk>
address varchar2(30)
country varchar2(20)
city varchar2(20)
building int
apartment int
sf_date_dimension
date_id bigint <pk>
day int(2)
month int(2)
year int(4)
full_date date
day_of_week varchar2(11)
quarter int(1)
st_employee_dimension
employee_id bigint <pk>
first_name varchar2(20)
last_name varchar2(20)
email varchar2(20)
phone varchar2(20)
restaurant_id varchar2(20)
job_title varchar2(20)
sf_employee_address_dimension
employee_address_id bigint <pk>
employee_id bigint <fk>
address varchar2(40)
country varchar(20)
city varchar2(20)
building int
apartment int
sf_department_dimension
department_id bigint <pk>
employee_id bigint <fk>
name <Undefined>

Figure 5.2