|  |
| --- |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status | [Alina](mailto:Kiryl_Bucha@epam.com) Makarets | 19-NOV-2017 |  |  |

Contents

[1. CREATE Storage Objects 3](#_Toc498882167)

[2. Generate Test Data in Storage Layers 3](#_Toc498882168)

[2.1. Анализ данных 3](#_Toc498882169)

[2.2. Создание пользователей и выделений grant на пользование таблицами, создание директории 4](#_Toc498882170)

[2.2.1. Пользователь sa\_src: 4](#_Toc498882171)

[2.2.2. Пользователь bl\_cl: 4](#_Toc498882172)

[2.2.3. Пользователь bl\_3nf: 4](#_Toc498882173)

[2.2.4. Создание директории 5](#_Toc498882174)

[2.2.5. Выделение грантов, с использованием пакетов: 5](#_Toc498882175)

[2.3. Создание external таблиц: 6](#_Toc498882176)

[2.4. Создание cleansing таблиц 6](#_Toc498882177)

[2.4.1. Скрипты для создания cleansing таблиц: 6](#_Toc498882178)

[2.5. Загрузка данных в cleansing таблицы 7](#_Toc498882179)

[2.5.1. Скрипты создания пакета и его тела 7](#_Toc498882180)

[2.5.2. Запуск пакета и проверка его работоспособности 8](#_Toc498882181)

[2.6. 3NF 9](#_Toc498882182)

[2.7. Конечная модель 11](#_Toc498882183)

[3. SQL\*Plus 11](#_Toc498882184)

# CREATE Storage Objects

# Generate Test Data in Storage Layers

## Анализ данных

Перед началом обработки данных и их загрузки в хранилище, нужно понять какие данные у нас есть сейчас и что они собой представляют. Для этого я посмотрела содержимое 3 прикрепленных к заданию файлов и провела анализ. Анализ проводился на примере страны Belarus

Содержимое файла iso\_3166.tab включает в себя информацию о странах, код страны в ISO формате, а также числовое поле, которое показывает уникальность страны (предположительно своеобразны id).



Рисунок 1 – Содержимое файла iso\_3166.tab

Содержимое файла iso\_3166\_geo\_un\_contries.tab включает в себя информацию о стране (id страны и ее название) и информацию о регионе, к которому она относится (id региона и название региона).

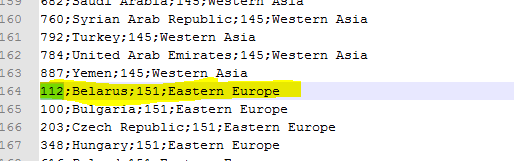


Рисунок 2 – Содержимое файла iso\_3166\_geo\_un\_contries.tab

Содержимое файла iso\_3166\_geo\_un.tab включает в себя информацию о название вселенной (world), названий континентов (Europe, Continents), и названий регионов (Eastern Europe;Regions).

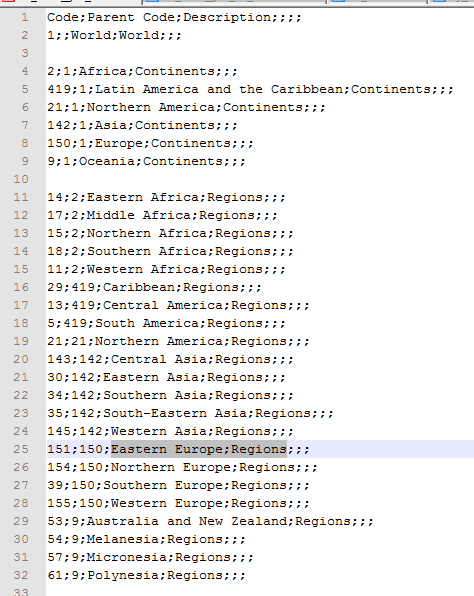


Рисунок 3 – Содержимое файла iso\_3166\_geo\_un.tab

Т.е. данный файл построил картинку об иерархии, которая представлена в данных трех файлах. Иерархия выглядит следующим образом :

World->Continents->Regions-Countries

## Создание пользователей и выделений grant на пользование таблицами, создание директории

Следующим шагом я создала 4 пользователя и выделила им некоторые гранты.

### Пользователь sa\_src:

CREATE USER sa\_src IDENTIFIED BY "123" DEFAULT TABLESPACE tbs\_pdb\_test;

GRANT CONNECT TO sa\_src;

GRANT RESOURCE TO sa\_src;

### Пользователь bl\_cl:

CREATE USER bl\_cl IDENTIFIED BY "123" DEFAULT TABLESPACE tbs\_pdb\_test;

GRANT CONNECT TO bl\_cl;

GRANT RESOURCE TO bl\_cl;

### Пользователь bl\_3nf:

CREATE USER bl\_3nf IDENTIFIED BY "123" DEFAULT TABLESPACE tbs\_pdb\_test;

GRANT CONNECT TO bl\_3nf;

GRANT RESOURCE TO bl\_3nf;

### Создание директории

Создадим объект типа директории, который будет хранить путь к external файлам:

CREATE DIRECTORY ext\_tables as 'C:\Users\Алина\Desktop\Topic 01 - ETL and SQL Review\';

В данном примере путь директории задался таким образом, потому что для выполнения данного задания я использовала домашний компьютер, который использует операционную систему WINDOWS.

### Выделение грантов, с использованием пакетов:

Выполним пакет для создания процедур на выдачу грантов.

CREATE OR REPLACE PACKAGE grants\_mgmt

AUTHID CURRENT\_USER

AS

PROCEDURE GRANT\_BLAT (GRANT\_NAME IN VARCHAR2,

SCHEMA\_NAME IN VARCHAR2,

OBJECT\_NAME IN VARCHAR2,

USER\_NAME IN VARCHAR2);

PROCEDURE GRANT\_BLAT (GRANT\_NAME IN VARCHAR2,

USER\_NAME IN VARCHAR2,

WAO IN BOOLEAN := FALSE);

END grants\_mgmt;

/

CREATE OR REPLACE PACKAGE BODY grants\_mgmt AS

PROCEDURE GRANT\_BLAT (GRANT\_NAME IN VARCHAR2,

SCHEMA\_NAME IN VARCHAR2,

OBJECT\_NAME IN VARCHAR2,

USER\_NAME IN VARCHAR2 )

IS

BEGIN

EXECUTE IMMEDIATE ('GRANT ' || GRANT\_NAME || ' ON ' || SCHEMA\_NAME || '.' || OBJECT\_NAME || ' TO ' || USER\_NAME);

END GRANT\_BLAT;

PROCEDURE GRANT\_BLAT (GRANT\_NAME VARCHAR2, USER\_NAME VARCHAR2, WAO IN BOOLEAN := FALSE) IS

BEGIN

IF WAO THEN

EXECUTE IMMEDIATE ('GRANT ' || GRANT\_NAME || ' TO ' || USER\_NAME || ' WITH ADMIN OPTION');

END IF;

EXECUTE IMMEDIATE ('GRANT ' || GRANT\_NAME || ' TO ' || USER\_NAME);

END;

END grants\_mgmt;

/

С использованием данного пакета дала недостающие гранты пользователям:

Пользователю SA\_SRC:

EXEC grants\_mgmt.grant\_blat('CREATE ANY TABLE', 'SA\_SRC');

EXEC grants\_mgmt.grant\_blat('SELECT ANY TABLE', 'SA\_SRC');

EXEC grants\_mgmt.grant\_blat('READ,WRITE ON DIRECTORY ext\_tables', 'SA\_SRC');

Пользователю BL\_CL:

EXEC grants\_mgmt.grant\_blat('CREATE ANY TABLE', 'BL\_CL');

EXEC grants\_mgmt.grant\_blat('SELECT ANY TABLE', 'BL\_CL');

EXEC grants\_mgmt.grant\_blat('READ,WRITE ON DIRECTORY ext\_tables', 'BL\_CL');

--Для выдачи данных грантов нужно заранее создать таблицы:

EXEC grants\_mgmt.grant\_blat ('SELECT', 'SRC','ext\_geo\_countries\_iso3166','BL\_CL');

EXEC grants\_mgmt.grant\_blat ('SELECT', 'SRC','ext\_geo\_structure\_iso3166','BL\_CL');

EXEC grants\_mgmt.grant\_blat ('SELECT', 'SRC','ext\_cntr2structure\_iso3166','BL\_CL');

Пользователю BL\_3NF:

EXEC grants\_mgmt.grant\_blat('CREATE ANY TABLE', 'BL\_3NF');

EXEC grants\_mgmt.grant\_blat('SELECT ANY TABLE', 'BL\_3NF');

--Для выдачи данных грантов нужно заранее создать таблицы:

EXEC grants\_mgmt.grant\_blat ('SELECT', 'BL\_CL','cl\_continents','BL\_3NF');

EXEC grants\_mgmt.grant\_blat ('SELECT', 'BL\_CL','cl\_countries','BL\_3NF');

EXEC grants\_mgmt.grant\_blat ('SELECT', 'BL\_CL','cl\_global','BL\_3NF');

EXEC grants\_mgmt.grant\_blat ('SELECT', 'BL\_CL','cl\_regions','BL\_3NF');

## Создание external таблиц:

Запустив скрипт, прилагавшийся к выполнению задания, мы получили 3 external таблицы. При этом таблицы создавались из под пользователя SA\_SRC:

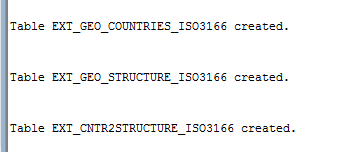


Рисунок 4 – Результат выполнения скрипта

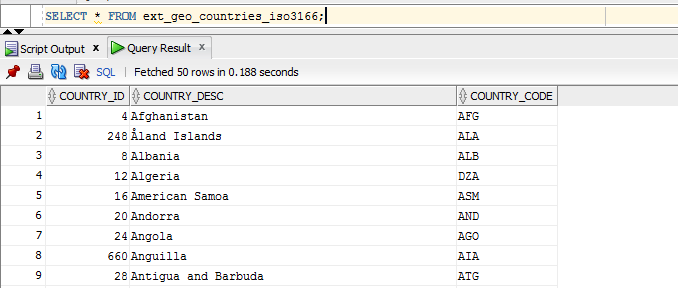


Рисунок 5 – Результат запроса из external таблицы

## Создание cleansing таблиц

На данном этапе создадим только cleansing тип таблиц, т.к. данных у нас относительно немного и чтение из external таблиц не будет занимать много времени. Но если бы у нас был большой объем данных, то создание work таблиц было бы обязательным условие для ускорение быстрого доступа к данным. Work таблицы создаются с такой же структурой, как и external таблицы, т.е. по такой же структуре, что и имеют наши данные.

Cleansing таблицы имеют структуру близкую к 3 нормальной форме. Исходя из выявленной выше иерархии, в нашей БД будет 4 таблицы:

* World
* Continents
* Regions
* Countries

### Скрипты для создания cleansing таблиц:

* World:

CREATE TABLE cl\_world

(

world\_id NUMBER(10,0),

world\_desc VARCHAR2 ( 200 CHAR )

);

* Continents:

CREATE TABLE cl\_continents

(

continent\_id NUMBER(10,0),

continent\_desc VARCHAR2 ( 200 CHAR ),

global\_id NUMBER(10,0)

);

* Regions

CREATE TABLE cl\_regions

(

region\_id NUMBER(10,0),

region\_desc VARCHAR2 ( 200 CHAR ),

continent\_id NUMBER(10,0)

);

* Countries

## Загрузка данных в cleansing таблицы

Загрузка данных из external таблиц в cleansing таблицы будет происходить с помощью пакета, в котором будет предусмотрена очистка данных путем нахождения определенного уровня.

### Скрипты создания пакета и его тела

Ниже представлен скрипт по созданию пакета для загрузки данных из external таблиц в cleansing таблицы.

Создание пакета:

CREATE OR REPLACE PACKAGE pkg\_etl\_geo

AS

PROCEDURE load\_worlds;

PROCEDURE load\_continents;

PROCEDURE load\_regions;

PROCEDURE load\_countries;

END pkg\_etl\_geo;

/

Создание тела пакета:

CREATE OR REPLACE PACKAGE body pkg\_etl\_geo

PROCEDURE load\_worlds

IS

BEGIN

EXECUTE immediate 'truncate table cl\_world';

INSERT INTO cl\_world

(world\_id,

world\_desc)

SELECT child\_code,

structure\_desc

FROM sa\_src.ext\_geo\_structure\_iso3166

WHERE structure\_level='World';

COMMIT;

EXCEPTION

WHEN OTHERS THEN

RAISE;

END load\_worlds;

PROCEDURE load\_continents

IS

BEGIN

EXECUTE immediate 'truncate table cl\_continents';

INSERT INTO cl\_continents

(continent\_id,

continent\_desc,

world\_id)

SELECT child\_code,

structure\_desc,

parent\_code

FROM src.ext\_geo\_structure\_iso3166

WHERE structure\_level='Continents';

COMMIT;

EXCEPTION

WHEN OTHERS THEN

RAISE;

END load\_continents;

PROCEDURE load\_regions

IS

BEGIN

EXECUTE immediate 'truncate table cl\_regions';

INSERT INTO cl\_regions

(region\_id,

region\_desc,

continent\_id)

SELECT child\_code,

structure\_desc,

parent\_code

FROM src.ext\_geo\_structure\_iso3166

WHERE structure\_level='Regions';

COMMIT;

EXCEPTION

WHEN OTHERS THEN

RAISE;

END load\_regions;

PROCEDURE load\_countries

IS

BEGIN

EXECUTE immediate 'truncate table cl\_countries';

INSERT INTO cl\_countries

(country\_id,

country\_desc,

country\_code,

region\_id)

SELECT country.country\_id,

country.country\_desc,

country.country\_code,

region.STRUCTURE\_CODE

FROM SA\_SRC.EXT\_GEO\_COUNTRIES\_ISO3166 country

JOIN SA\_SRC.EXT\_CNTR2STRUCTURE\_ISO3166 region

ON country.COUNTRY\_ID=region.COUNTRY\_ID;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

RAISE;

END load\_countries;

END pkg\_etl\_geo;

/

### Запуск пакета и проверка его работоспособности

Для запуска пакета необходимо запустить следующие скрипты:

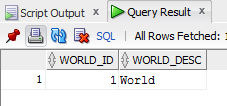
EXEC PKG\_ETL\_GEO.load\_worlds;

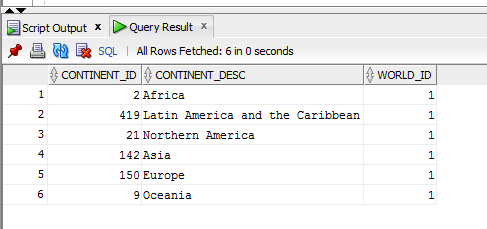
EXEC PKG\_ETL\_GEO.load\_continents;

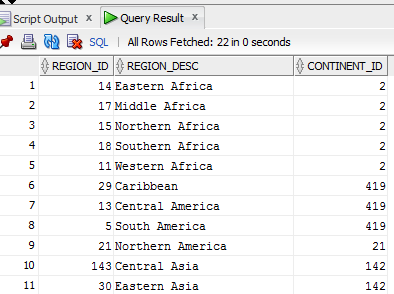
EXEC PKG\_ETL\_GEO.load\_regions;

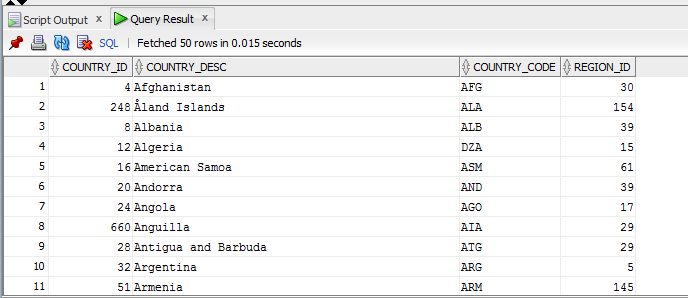
EXEC PKG\_ETL\_GEO.load\_countries;

Проверим правильность выполнения пакета:









## 3NF

На данном этапе нам необходимо создать основные таблицы. На данном этапе необходимо продумать связи с таблицами, т.е. ограничения. Ниже представлены скрипты на их создания.

CREATE TABLE ce\_worlds

(

world\_id NUMBER(10,0) PRIMARY KEY ,

world\_desc VARCHAR2 ( 200 CHAR ) NOT NULL

);

CREATE TABLE ce\_regions

(

region\_id NUMBER(10,0) PRIMARY KEY,

region\_desc VARCHAR2 ( 200 CHAR ) NOT NULL,

continent\_id NUMBER(10,0) NOT NULL,

CONSTRAINT FK\_continent\_id FOREIGN KEY (continent\_id)

REFERENCES ce\_continents(continent\_id)

);

CREATE TABLE ce\_countries

(

country\_id NUMBER(10,0) PRIMARY KEY NOT NULL,

country\_desc VARCHAR2 ( 200 CHAR ) NOT NULL,

country\_code VARCHAR2 ( 3 ),

region\_id NUMBER(10,0)NOT NULL ,

CONSTRAINT FK\_country\_id FOREIGN KEY (region\_id)

REFERENCES ce\_regions(region\_id)

);

CREATE TABLE ce\_continents

(

continent\_id NUMBER(10,0) PRIMARY KEY ,

continent\_desc VARCHAR2 ( 200 CHAR ) NOT NULL,

world\_id NUMBER(10,0) NOT NULL,

CONSTRAINT FK\_world\_id FOREIGN KEY (world\_id)

REFERENCES ce\_worlds (world\_id)

);

Теперь проведем вставку данных:

INSERT INTO ce\_worlds

(world\_id,

world\_desc)

SELECT world\_id,

world\_desc

FROM bl\_cl.cl\_world;

INSERT INTO ce\_continents

(continent\_id,

continent\_desc,

world\_id)

SELECT continent\_id,

continent\_desc,

world\_id

FROM bl\_cl.cl\_continents;

INSERT INTO ce\_regions

(region\_id,

region\_desc,

continent\_id)

SELECT region\_id,

region\_desc,

continent\_id

FROM bl\_cl.cl\_regions;

INSERT INTO ce\_countries

(country\_id,

country\_desc,

country\_code,

region\_id)

SELECT country\_id,

country\_desc,

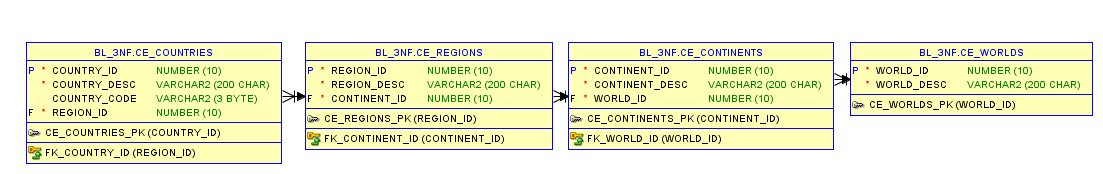
country\_code,

region\_id

FROM bl\_cl.cl\_countries;

## Конечная модель

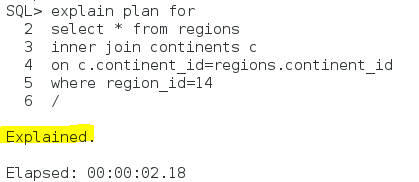
Конечная модель данных приведена ниже:

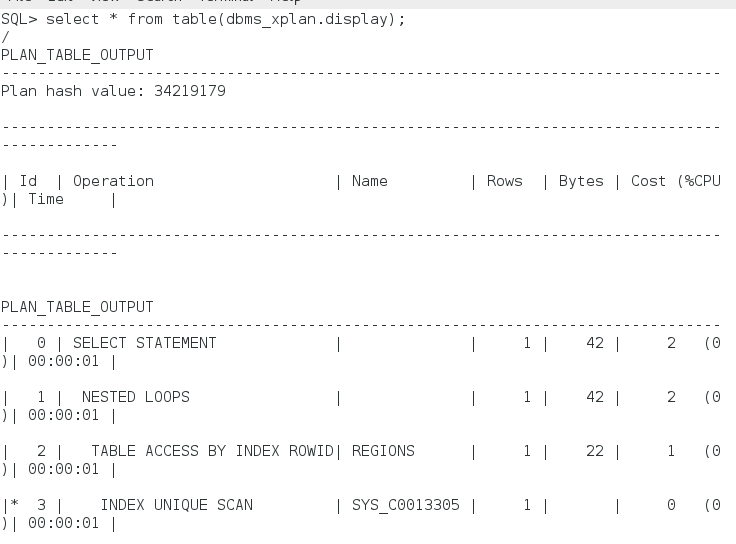


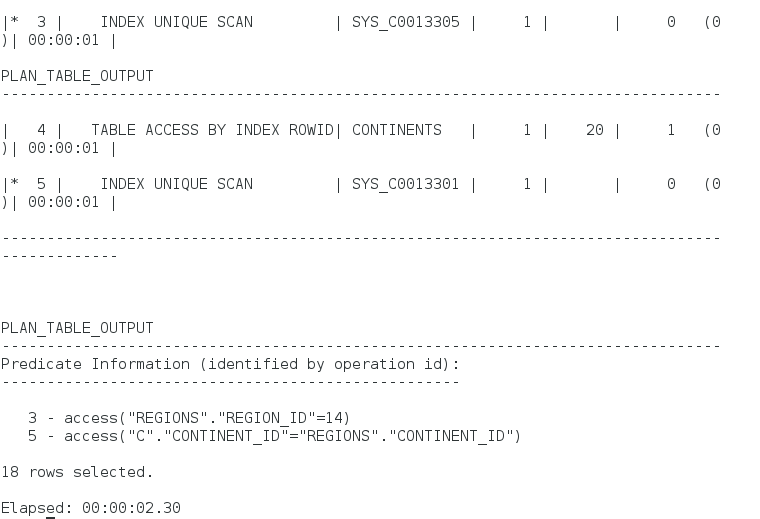
# SQL\*Plus

Connect to a database via SQL\*Plus client and do next steps:

## Show execution plans in SQL\*Plans



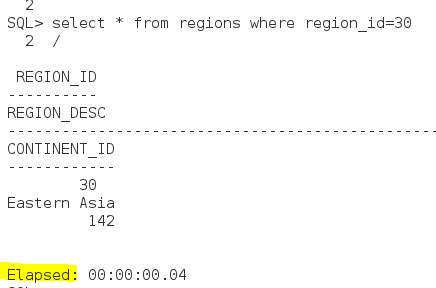




## Set timing on



## Run script



## Save data to file

