

**EPAM Systems, RD Dep.**

# **Core PL/SQL**

---

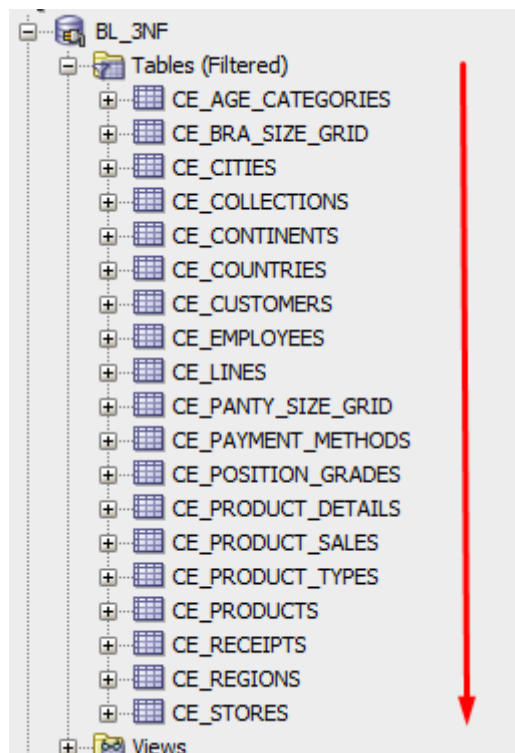
REVISION HISTORY					
Ver.	Description of Change	Author	Date	Approved	
				Name	Effective Date
1.0	Initial status	<a href="#">Valeryia_Lupanava</a>	25-NOV-2017		

## Содержание

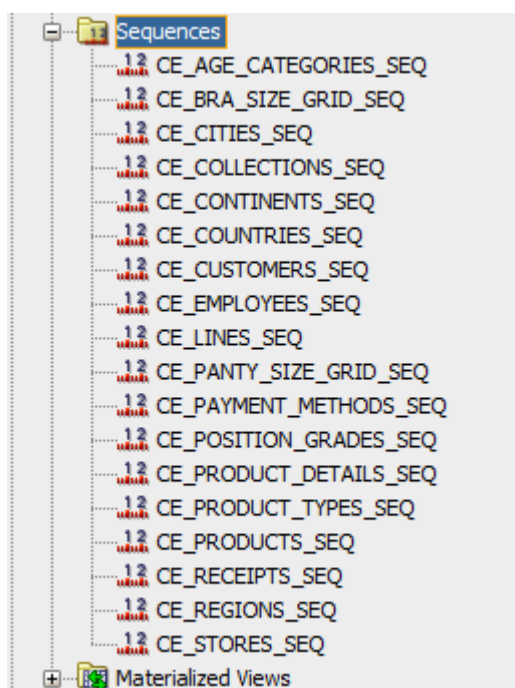
1. СОЗДАНИЕ ОБЪЕКТОВ 3NF-СЛОЯ .....	3
2. ВЫДАЧА ГРАНТОВ CL-СЛОЮ .....	5
3. ПАКЕТЫ ДЛЯ ЗАГРУЗКИ 3NF-СЛОЯ.....	6

## 1. Создание объектов 3NF-слоя

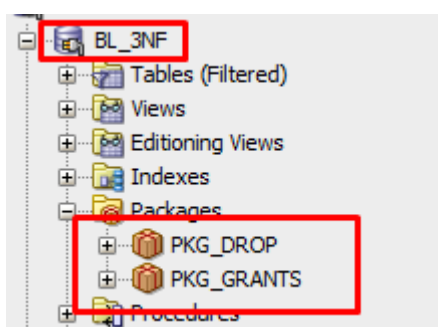
- На 3NF-слое были созданы следующий объекты.



- Диаграмму 3NF-слоя находится в папке с данным отчетом.
- Также на этом слое были созданы все необходимые sequence.



- На этом слое дополнительно были использованы пакеты на выдачу грантов и на удаление объектов.



pkg_drop_tables.sql	11/22/2017 3:45 PM	Microsoft SQL Ser...	2 KB
pkg_grants.sql	11/18/2017 12:41 ...	Microsoft SQL Ser...	2 KB

- Структура папок 3NF-схемы имеет следующий вид. Для каждого объекта создана папка с соответствующим названием. В папке располагается скрипт на создание объекта(1) и скрипт на создание sequence(2). Скрипты на заполнение объектов располагаются в релевантных пакетах для заполнения в BL\_CL-слое, однако скрипт на запуск заполнения объекта(3) расположен в папке объекта на 3NF-слое.

ce_age_categories.sql	1	11/25/2017 1:29 PM	Microsoft SQL Ser...	1 KB
ce_age_categories_insert.sql	2	11/25/2017 1:33 PM	Microsoft SQL Ser...	1 KB
ce_age_categories_sequence.sql	3	11/25/2017 1:46 AM	Microsoft SQL Ser...	1 KB

- Проверка после заполнения данных в 3NF-слое.

	SALE_ID	RECEIPT_SRCID	PRODUCT_DETAILS_SRCID	SALE_SUM_USD	SALE_AMOUNT	INSERT_DT	RECEIPT_ID	RECEIPT_SRCID_1	RECEIPT_NUMBER	RECEIPT_DT
1	990	1300398	39227	1483650	476	25-NOV-17	98425	1300398	670535264025	29-OCT-19
2	991	1287317	36127	2963872	597	25-NOV-17	85344	1287317	2069295089269	20-JUL-18
3	992	1248054	39138	5931770	476	25-NOV-17	46081	1248054	2897971074870	09-NOV-18
4	993	1240426	38560	4426873	942	25-NOV-17	38453	1240426	3429884592980	25-JAN-19
5	994	1264990	38056	1835483	786	25-NOV-17	63017	1264990	6516696253041	10-NOV-18
6	995	1270326	40535	5351632	433	25-NOV-17	68353	1270326	2151341922882	19-SEP-18
7	996	1287526	39387	8343738	106	25-NOV-17	85553	1287526	561044322287	20-MAR-19
8	997	1295069	36916	2838508	114	25-NOV-17	93096	1295069	9850913310038	29-OCT-19
9	998	1262681	37000	4716432	152	25-NOV-17	60708	1262681	2969697299134	13-JUL-19
10	999	1217937	41799	4587870	657	25-NOV-17	15964	1217937	8133611806129	24-MAY-19
11	1000	1274437	38482	8765778	652	25-NOV-17	72464	1274437	7095657711554	10-SEP-18
12	1001	1229457	40187	2869036	577	25-NOV-17	27484	1229457	5055480645091	17-DEC-19
13	1002	1299862	36434	1767338	849	25-NOV-17	97889	1299862	1359252188665	05-OCT-19
14	1003	1246954	41947	5938174	803	25-NOV-17	44981	1246954	5613177890287	25-MAR-18

## 2. Выдача грантов CL-слою

- Поскольку заполнение объектов происходило их BL\_CL-слоя, были выданы гранты:
  - INSERT ce-table – при заполнении таблицы данными выполняется INSERT;
  - SELECT ce-table – при merge-операции необходимо сначала сделать выборку данных из ce-table для проверки на дубли перед заполнением;
  - SELECT ce-sequence – при merge-операции в скрипте используется sequence для создания суррогатного ключа;
  - UPDATE ce-table – при merge-операции, если попадает значение, которое ранее было добавлено, осуществляется UPDATE.

```
BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'INSERT', SCHEMA_NAME => 'BL_3NF', OBJECT_NAME => 'CE_PRODUCT_SALES', USER_NAME => 'BL_CL_1ST');
END;

BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'SELECT', SCHEMA_NAME => 'BL_3NF', OBJECT_NAME => 'CE_PRODUCT_SALES', USER_NAME => 'BL_CL_1ST');
END;

BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'SELECT', SCHEMA_NAME => 'BL_3NF', OBJECT_NAME => 'CE_PRODUCT_SALES_SEQ', USER_NAME => 'BL_CL_1ST');
END;

BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'UPDATE', SCHEMA_NAME => 'BL_3NF', OBJECT_NAME => 'CE_PRODUCT_SALES', USER_NAME => 'BL_CL_1ST');
END;
```

- Для каждого объекта последовательно выдавались соответствующие гранты. Гранты на операцию с любым объектом схемы не применялись.

### 3. Пакеты для загрузки 3NF-слоя

- Пакеты на загрузку данных в 3NF-слой расположены в BL\_CL-слое. Процедуры загрузки для объектов распределены по пакетам в соответствии с предметной областью.
- **Explicit Cursor.**

Скрипт.

```
CREATE OR REPLACE PROCEDURE explicit_cursor
IS
BEGIN
DECLARE
    CURSOR c_data IS
        SELECT A.continent_id,
               b.continent_desc
        FROM cls_continents A
             LEFT OUTER JOIN bl_3nf.ce_continents b
                 ON A.continent_id = b.continent_srcid;
TYPE t_data IS TABLE OF c_data%rowtype INDEX BY BINARY_INTEGER;
t_data t_data;
BEGIN
    OPEN c_data;
    LOOP
        FETCH c_data BULK COLLECT INTO t_data;
        EXIT WHEN t_data.COUNT = 0;
        FOR idx IN t_data.FIRST .. t_data.LAST LOOP
            IF t_data(idx).continent_desc IS NULL THEN
                INSERT INTO bl_3nf.ce_continents
                SELECT bl_3nf.ce_continents_seq.NEXTVAL,
                       continent_id,
                       continent_desc
                FROM (
                    (SELECT continent_id,
                             continent_desc
                     FROM cls_continents
                     MINUS
                     SELECT A.continent_id,
                            A.continent_desc
                     FROM cls_continents A,
                          bl_3nf.ce_continents b
                     WHERE A.continent_id = b.continent_srcid));
            ELSE
                UPDATE bl_3nf.ce_continents
                SET continent_desc = t_data(idx).continent_desc;
            END IF;
        END LOOP;
    END LOOP;
    CLOSE c_data;
```

```

END;
END;
/
BEGIN
explicit_cursor;
END;

```

```

DECLARE 1
CURSOR c_data IS
SELECT A.continent_id, 2
       b.continent_desc
FROM   cls_continents A
       LEFT OUTER JOIN bl_3nf.ce_continents b
       ON A.continent_id = b.continent_srcid;
TYPE t_data IS TABLE OF c_data%rowtype INDEX BY BINARY_INTEGER; 3
t_data t_data;

```

1. Создаем курсор C\_DATA;
2. Определяем данные для заполнения;
3. Создаем таблицу T\_\_DATA, в которую будем помещать значения нашего курсора C\_DATA. Для C\_DATA выбрано тип ROWTYPE, поскольку считывается не одно значение колонки, а строка из двух значений. Для таблицы определяем переменную T\_DATA, с помощью которой будем считывать строки из T\_\_DATA.

```

FOR idx IN t_data.FIRST .. t_data.LAST LOOP
IF t_data(idx).continent_desc IS NULL THEN 1
INSERT INTO bl_3nf.ce_continents
SELECT bl_3nf.ce_continents_seq.NEXTVAL,
       continent_id,
       continent_desc
FROM (
(SELECT continent_id,
        continent_desc 2
FROM   cls_continents
MINUS
SELECT A.continent_id,
        A.continent_desc
FROM   cls_continents A,
        bl_3nf.ce_continents b
WHERE  A.continent_id = b.continent_srcid));
ELSE
UPDATE bl_3nf.ce_continents 3
SET    continent_desc = t_data(idx).continent_desc;
END IF;
END LOOP;

```

1. Проверяем, если CONTINENT\_DESC, переданный в T\_DATA, равен нулю (поскольку применен LEFT OUTER JOIN, то континенты из 3NF будут либо равны NULL, либо в T\_DATA будет передано значение), то есть такого значения нет еще в CE\_CONTINENTS, то вставляем данные, если есть, то просто обновляем.
2. INSERT, если данных нет.

## 3. UPDATE, если данные есть.

```

END LOOP;
END LOOP;
CLOSE c_data;
END;
END;

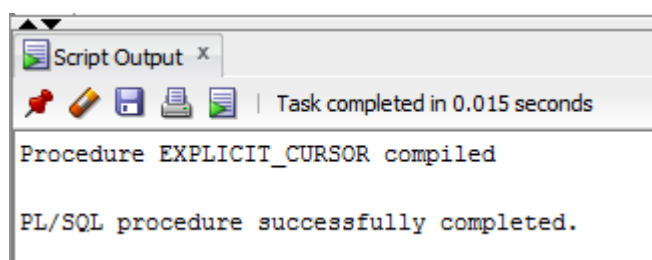
```

- Обязательно закрываем курсор в конце.
- Вызываем процедуру с курсором для проверки.

```

BEGIN
explicit_cursor;
END;

```



- **Implicit Cursor.**

Скрипт.

```

CREATE OR REPLACE PROCEDURE implicit_cursor
IS
BEGIN
DECLARE
CURSOR my_cursor IS
SELECT a.continent_id as continent_id,
       a.continent_desc as continent_desc
FROM   cls_continents a
       LEFT OUTER JOIN BL_3NF.CE_CONTINENTS b
                   ON a.continent_id = b.CONTINENT_SRCID
WHERE  b.CONTINENT_SRCID IS NULL;
BEGIN
FOR my_cursor_val IN my_cursor
LOOP
INSERT INTO BL_3NF.CE_CONTINENTS (continent_id,
                                  continent_srcid,
                                  continent_desc)
VALUES (BL_3NF.ce_continents seq.NEXTVAL,
        my_cursor_val.continent_id,
        my_cursor_val.continent_desc);

END LOOP;
COMMIT;
END;

```



```
END;
/
begin
implicit_cursor;
end;
```

- В данном типе курсора не требуется явное открытие и закрытие курсора. Достаточно объявить курсор, передав в него, например, SELECT.

```

BEGIN
DECLARE
CURSOR my_cursor IS
SELECT A.continent_id AS continent_id,
       A.continent_desc AS continent_desc
FROM   cls_continents A
       LEFT OUTER JOIN bl_3nf.ce_continents b
         ON A.continent_id = b.continent_srcid
WHERE  b.continent_srcid IS NULL;
BEGIN
```

- В данном SELECT сразу передавались значения, которых нет в конечной таблице.
- Поскольку мы определили сразу данные для INSERT, то можно переходить к их добавлению, последовательно передавая данные из курсора в цикле.

```

FOR my_cursor_val IN my_cursor
LOOP
INSERT INTO bl_3nf.ce_continents (continent_id,
                                continent_srcid,
                                continent_desc)
VALUES (bl_3nf.ce_continents_seq.NEXTVAL,
        my_cursor_val.continent_id,
        my_cursor_val.continent_desc);
END LOOP;
```

- Вызываем процедуру с курсором для проверки.

```

/
BEGIN
implicit_cursor;
END;
```

```

Procedure IMPLICIT_CURSOR compiled
PL/SQL procedure successfully completed.
```

- Merge.**

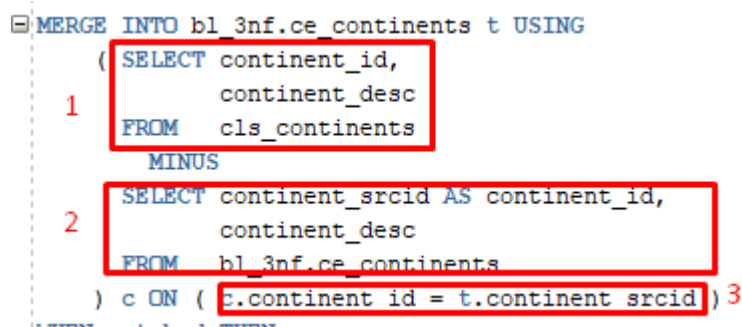
Скрипт.

```

PROCEDURE merge_table_ce_continents
IS
```

```
BEGIN
MERGE INTO bl_3nf.ce_continents t USING
  ( SELECT continent_id,
          continent_desc
    FROM cls_continents
      MINUS
      SELECT continent_srcid AS continent_id,
          continent_desc
    FROM bl_3nf.ce_continents
  ) c ON ( c.continent_id = t.continent_srcid )
WHEN matched THEN
  UPDATE SET t.continent_desc = c.continent_desc
WHEN NOT matched THEN
  INSERT
  (
    continent_ID ,
    continent_SRCID ,
    continent_desc
  )
  VALUES
  (
    bl_3nf.ce_continents_seq.NEXTVAL,
    c.continent_ID ,
    c.continent_desc
  ) ;
COMMIT;
EXCEPTION
WHEN OTHERS THEN
  RAISE;
END merge_table_ce_continents;
```

- При использовании MERGE есть следующие шаги.



The screenshot shows a portion of the SQL code from the previous block, specifically the MERGE statement. Three parts of the code are highlighted with red boxes and numbered:

- 1. The first SELECT query: `SELECT continent_id, continent_desc FROM cls_continents`
- 2. The second SELECT query: `SELECT continent_srcid AS continent_id, continent_desc FROM bl_3nf.ce_continents`
- 3. The ON clause: `c.continent_id = t.continent_srcid`

1. Определили совокупный набор данных.
2. Определили тот набор данных, который есть в итоговой таблице.

3. С помощью MINUS удалили из первого набора второй. В результате получаем набор для INSERT.
- Далее полученный набор сравниваем полностью с итоговой таблицей для заполнения по ключевому полю CONTINENT\_ID. Если совпадение есть, то происходит просто обновление строки в итоговой таблице.

```
WHEN matched THEN  
    UPDATE SET t.continent_desc = c.continent_desc
```

- Если совпадения нет, то осуществляется INSERT.

```
WHEN NOT matched THEN  
    INSERT  
    (  
        continent_ID ,  
        continent_SRCID ,  
        continent_desc  
    )  
    VALUES  
    (  
        bl_3nf.ce_continents_seq.NEXTVAL,  
        c.continent_ID ,  
        c.continent_desc  
    ) ;
```

```
Procedure MERGE_TABLE_CE_CONTINENTS compiled
```

```
PL/SQL procedure successfully completed.
```