Saved: 26-Nov-2017 21:02

EPAM Systems, RD Dep.

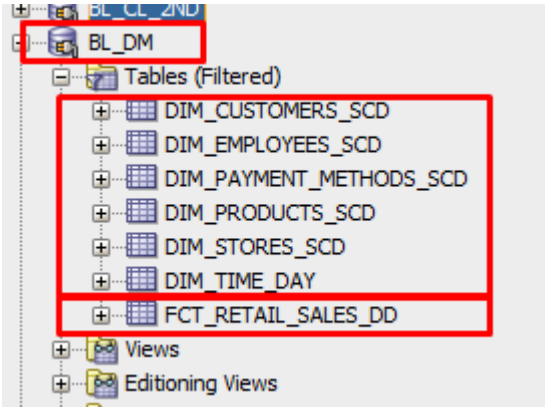
Advanced PL/SQL

REVISION HISTORY					
Ver.	**Description of Change**	**Author**	**Date**	**Approved**	
				Name	**Effective Date**
1.0	Initial status	Valeryia_Lupanava	26-NOV-2017		

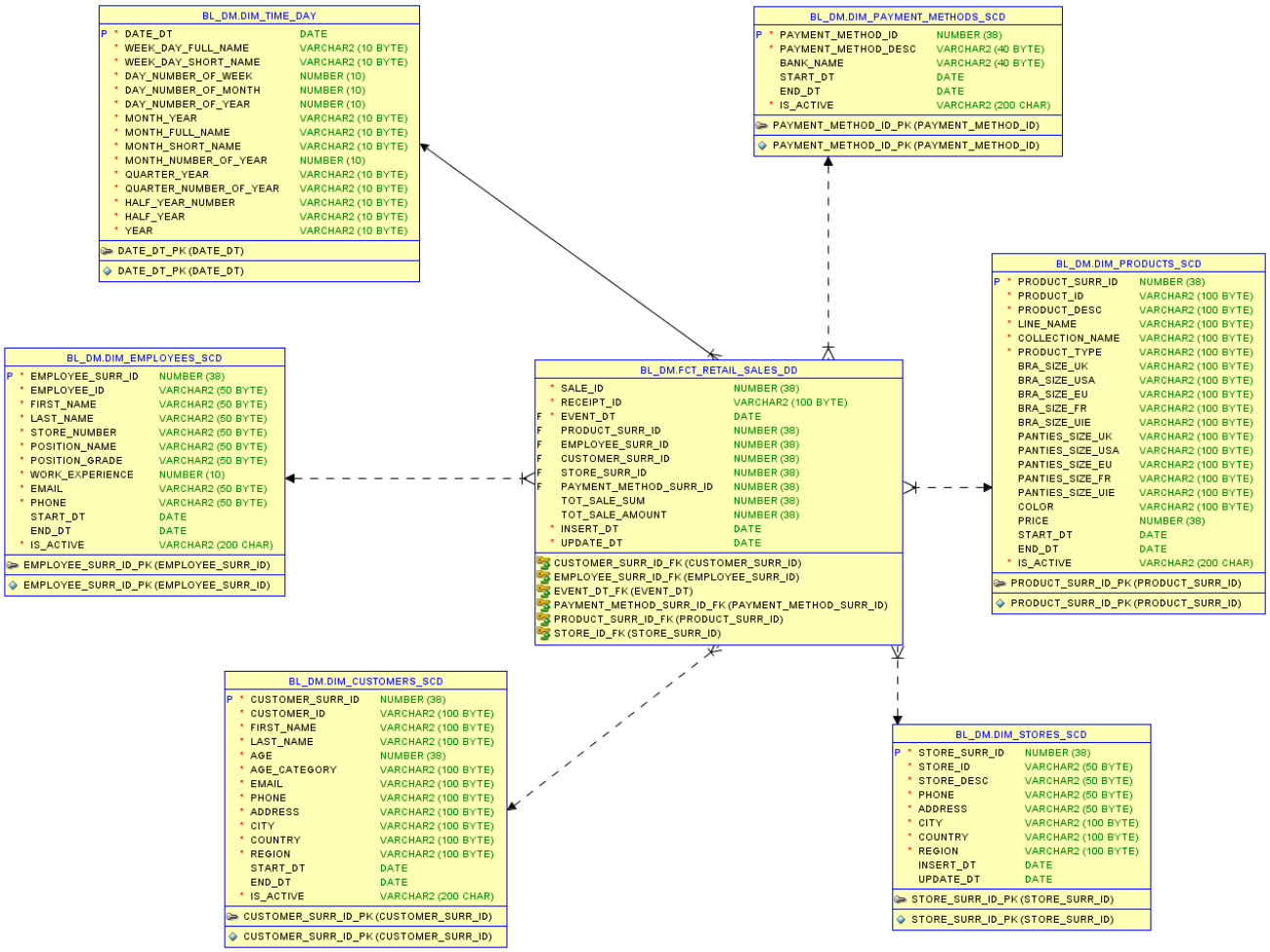
Содержание

1. Создание таблиц для DM-слоя

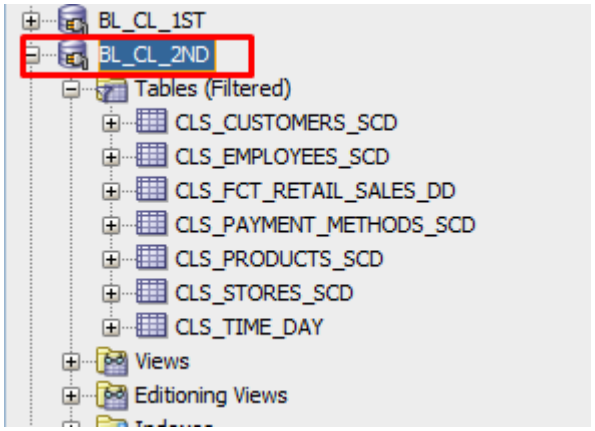
- На DM-слое были создано пять дименшен-таблиц и одна фактовая таблица. Сразу были определены все CONSTRAINTS.



- Схема DM-слоя. Для DM-слоя, как видно, была выбрана схема STAR.



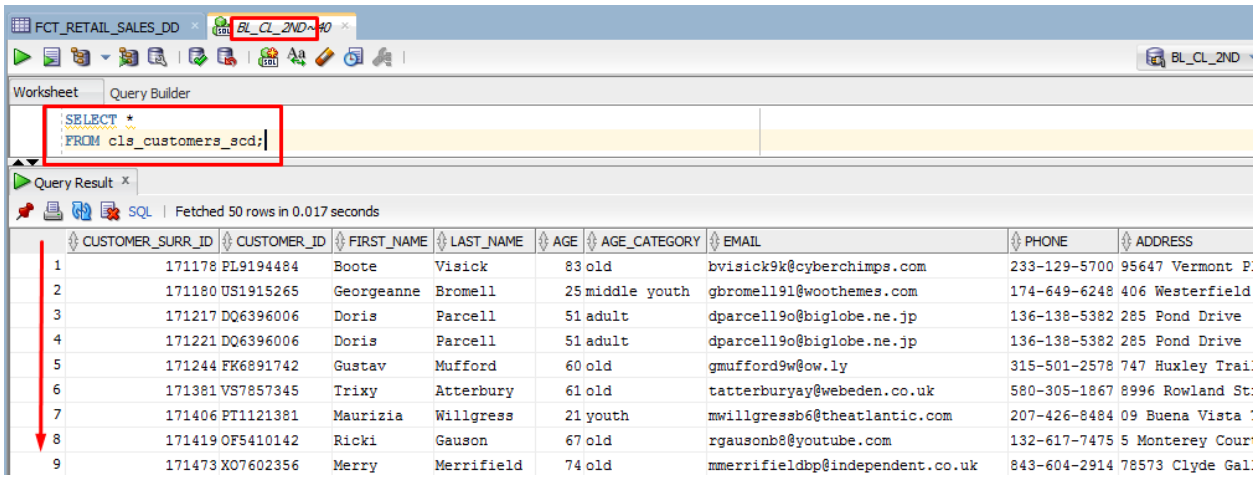
- Следующим шагом были созданы таблицы на втором клинзинговом слое BL_CL_2ND, повторяющие структуру таблиц DM-слоя.



- Изначально были заполнены объекты BL_CL_2ND- слоя. При заполнении данного слоя были сагрегированы таблицы 3NF-слоя.
- Таблицы BL_CL_2ND и BL_DM слоев загружались с помощью пакетов, которые можно найти по директории **…\dwso\bl_cl_2nd\packages.**
- Пример скрипта на заполнение таблицы BL_CL_2ND-слоя.

```
TRUNCATE TABLE cls_customers_scd;
  INSERT INTO cls_customers_scd
  SELECT DISTINCT
           customer_srcid AS customer_surr_id,
           customer_number AS customer_id,
           first_name,
           last_name,
           age,
           ac.age_category_desc AS age_category,
           email,
           phone,
           address,
           cs.city_desc AS city,
           cn.country_desc AS country,
           cr.region_desc AS region,
           start_dt,
           end_dt,
           is_active
  FROM   bl_3nf.ce_customers cc left join
bl_3nf.ce_age_categories ac
                      on cc.age_category_srcid
=c.age_category_srcid
                              left join bl_3nf.ce_cities cs
                      on cc.city_srcid = cs.city_srcid
                              left join bl_3nf.ce_countries cn
                      on cs.country_srcid = cn.country_srcid
                              left join bl_3nf.ce_regions cr
                      on cn.region_srcid = cr.region_srcid;
COMMIT;
```

- Результат.



SELECT *
FROM cls_customers_scd;

Query Result | SQL | Fetched 50 rows in 0.017 seconds

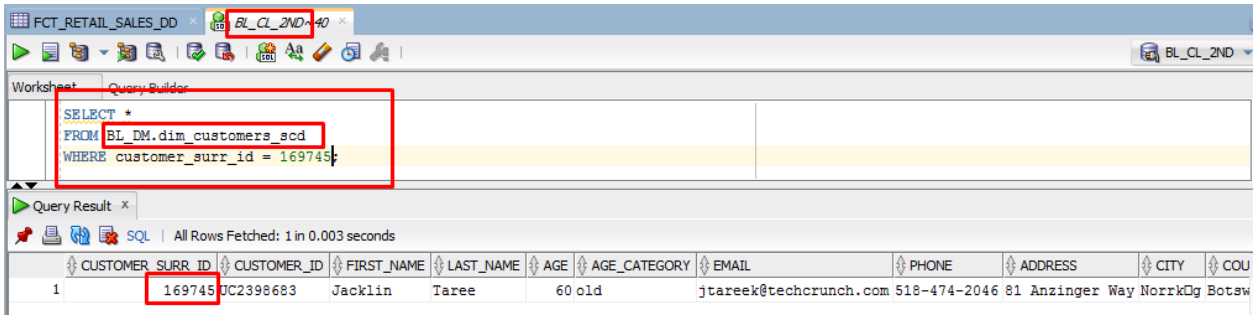
	CUSTOMER_SURR_ID	CUSTOMER_ID	FIRST_NAME	LAST_NAME	AGE	AGE_CATEGORY	EMAIL	PHONE	ADDRESS
1	171178	PL9194484	Boote	Visick	83	old	bvisick9k@cyberchimps.com	233-129-5700	95647 Vermont P
2	171180	US1915265	Georgeanne	Bromell	25	middle youth	gbromell91@woothemes.com	174-649-6248	406 Westerfield
3	171217	DQ6396006	Doris	Parcell	51	adult	dparcell9o@biglobe.ne.jp	136-138-5382	285 Pond Drive
4	171221	DQ6396006	Doris	Parcell	51	adult	dparcell9o@biglobe.ne.jp	136-138-5382	285 Pond Drive
5	171244	FK6891742	Gustav	Mufford	60	old	gmufford9w@ow.ly	315-501-2578	747 Huxley Trai
6	171381	VS7857345	Trixy	Atterbury	61	old	tatterburyay@webeden.co.uk	580-305-1867	8996 Rowland St
7	171406	PT1121381	Maurizia	Willgress	21	youth	mwillgressb6@theatlantic.com	207-426-8484	09 Buena Vista
8	171419	OF5410142	Ricki	Gauson	67	old	rgausonb8@youtube.com	132-617-7475	5 Monterey Cour
9	171473	XO7602356	Merry	Merrifield	74	old	mmerrifieldbp@independent.co.uk	843-604-2914	78573 Clyde Gal

- Пример скрипта на заполнение таблицы BL_DM-слоя.

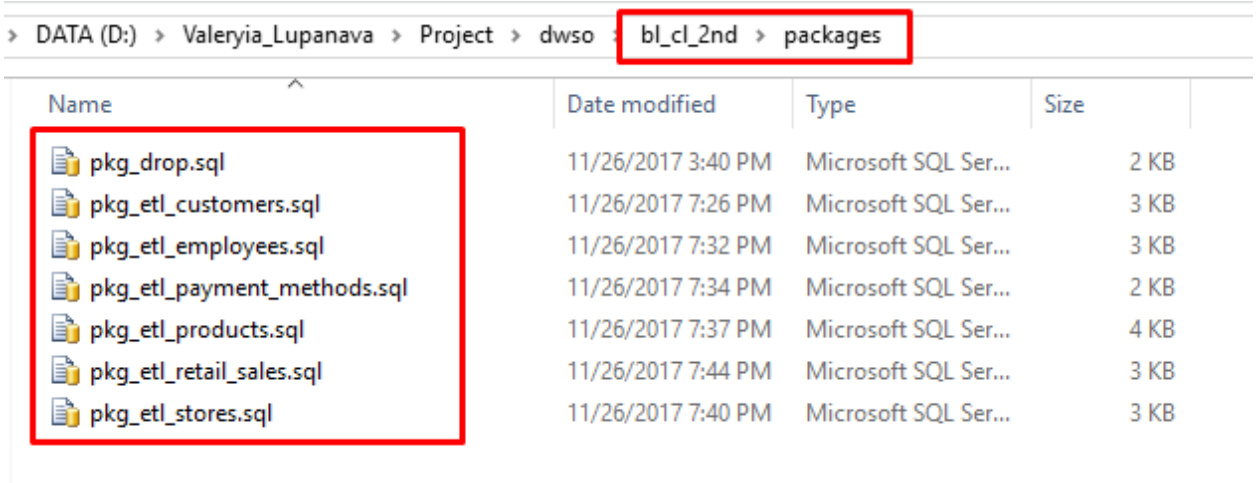
```
MERGE INTO bl_dm.dim_customers_scd t USING
    ( SELECT *
      FROM   cls_customers_scd
        MINUS
      SELECT *
      FROM   bl_dm.dim_customers_scd
    ) c ON ( c.customer_surr_id = t.customer_surr_id )
WHEN matched THEN
      UPDATE SET t.customer_id = c.customer_id
WHEN NOT matched THEN
      INSERT
      (
       customer_surr_id,
       customer_id,
       first_name,
       last_name,
       age,
       age_category,
       email,
       phone,
       address,
       city,
       country,
       region,
       start_dt,
       end_dt,
       is_active
      )
      VALUES
      (
       c.customer_surr_id,
       c.customer_id,
       c.first_name,
       c.last_name,
       c.age,
```

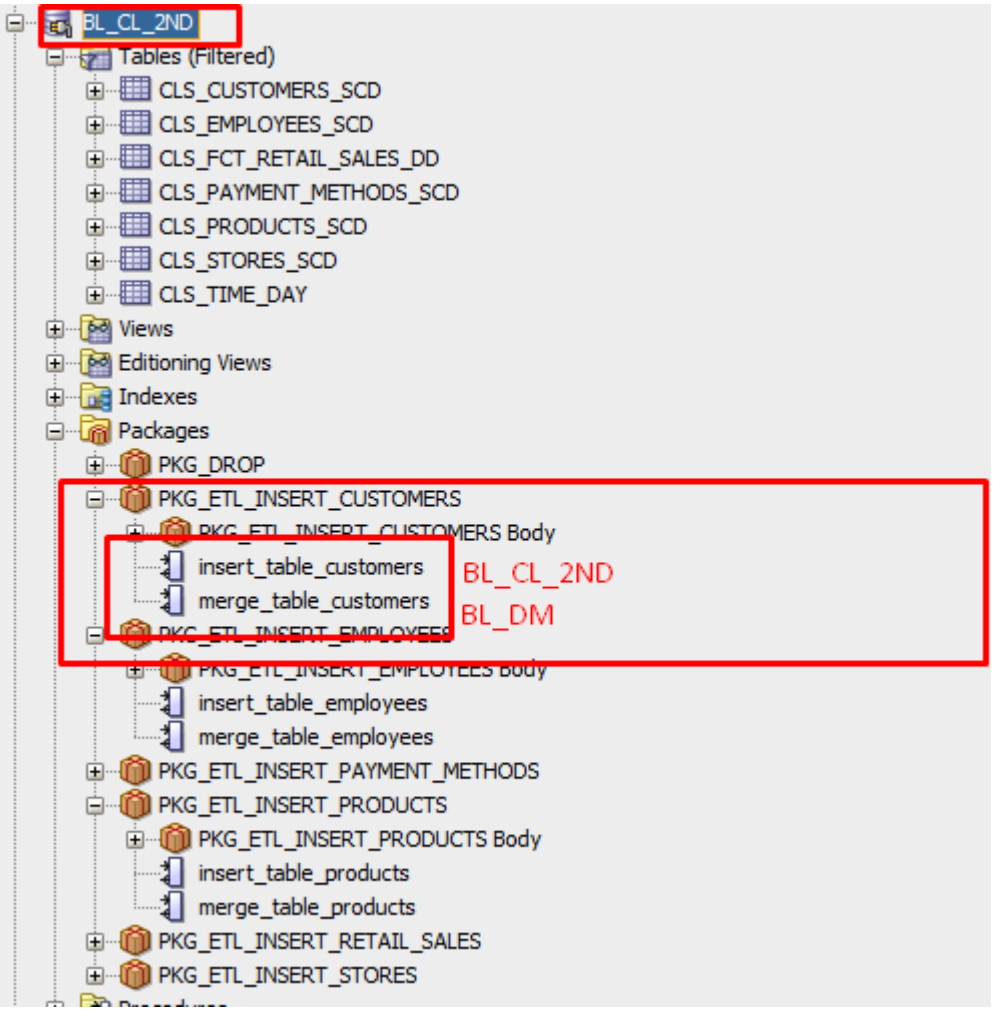
```
        c.age_category,
        c.email,
        c.phone,
        c.address,
        c.city,
        c.country,
        c.region,
        c.start_dt,
        c.end_dt,
        c.is_active
      ) ;
COMMIT;
```

- Результат.



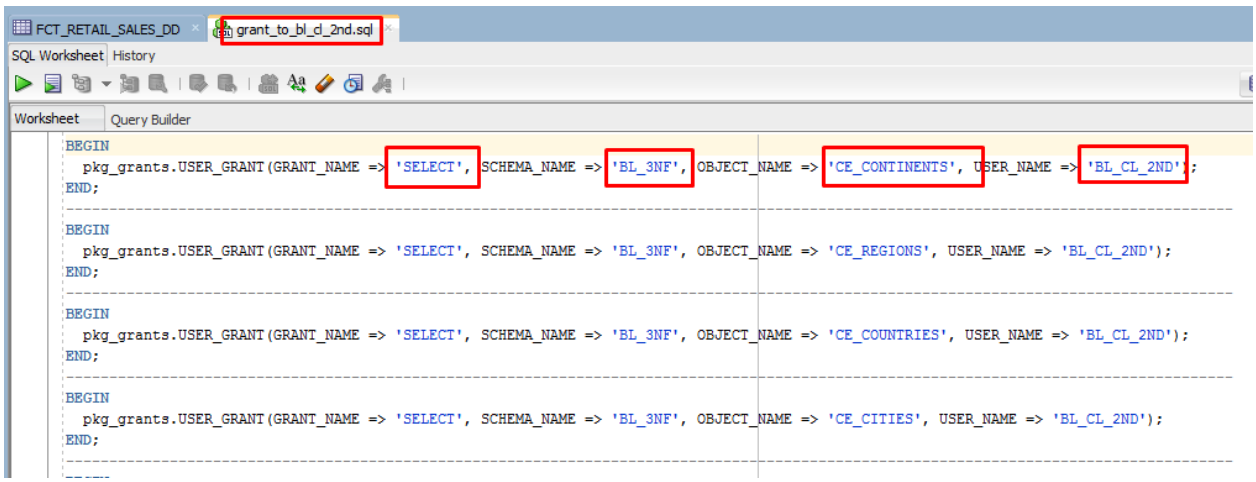
- Все процедуры на загрузку BL_CL_2ND и BL_DM слоев организованы в пакетах на BL_CL_2ND-слое. Для этого BL_CL_2ND – слою были выданы все необходимы гранты из BL_3NF и BL_DM слоев. Описание выданных грантов описано в следующей главе.



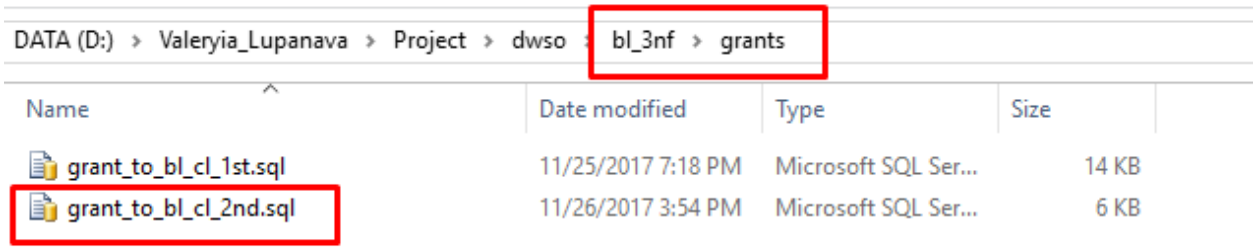


2. Выдача грантов CL-слою

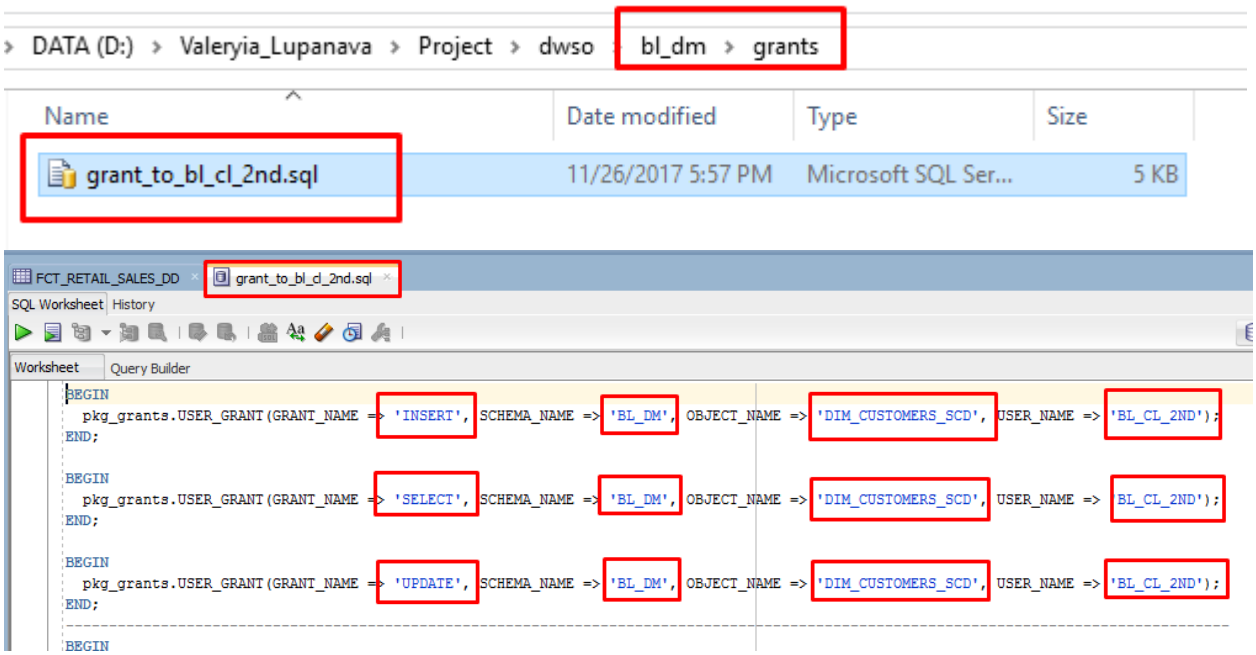
- Для заполнения BL_CL_2ND и BL_DM слоев BL_CL_2ND-слою понадобились следующие гранты. Из BL_3NF слою BL_CL_2ND были выданы гранты на SELECT из соответствующих таблиц.



- Файл с грантами расположен в папке BL_3NF.



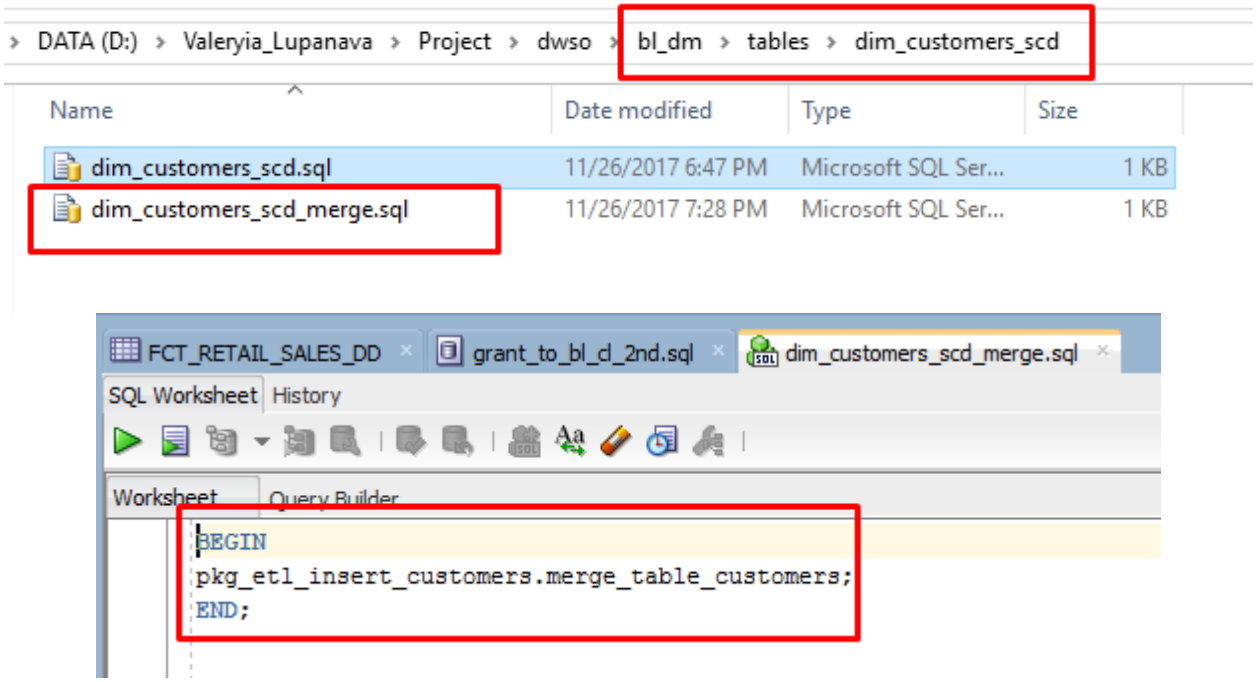
- Также BL_CL_2ND были выданы гранты из BL_DM слоя для заполнения соответствующих дименшен-таблиц и фактовой таблицы.



- Было использовано три типа грантов: INSERT, UPDATE, SELECT.
  - 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;
  - UPDATE ce-table – при merge-операции, если попадается значение, которое ранее было добавлено, осуществляется UPDATE.

3. Создание пакетов для заполнения DM-слоя

- Для заполнения объектов данного слоя, как было описано выше, использовалась пакетная загрузка.
- Запуска пакета осуществлялся из BL_CL_2ND – слоя. Однако, скрипты на запуск пакетов расположены в папках соответствующих DDL-объектов.



- Для заполнения объектов данного DM-слоя использовалась операция MERGE, поскольку все таблицы с CONSTRAINTS.
- Пример скрипта.

```
MERGE INTO bl_dm.dim_customers_scd t USING
    ( SELECT *
      FROM   cls_customers_scd
        MINUS
      SELECT *
      FROM   bl_dm.dim_customers_scd
    ) c ON ( c.customer_surr_id = t.customer_surr_id )
WHEN matched THEN
      UPDATE SET t.customer_id = c.customer_id
WHEN NOT matched THEN
      INSERT
      (
       customer_surr_id,
       customer_id,
       first_name,
       last_name,
       age,
       age_category,
       email,
       phone,
       address,
       city,
       country,
       region,
       start_dt,
       end_dt,
       is_active
```

```
      )
      VALUES
      (
       c.customer_surr_id,
       c.customer_id,
       c.first_name,
       c.last_name,
       c.age,
       c.age_category,
       c.email,
       c.phone,
       c.address,
       c.city,
       c.country,
       c.region,
       c.start_dt,
       c.end_dt,
       c.is_active
      ) ;
COMMIT;
```

- При использовании MERGE есть следующие шаги.

```
BEGIN
  MERGE INTO bl_dm.dim_customers_scd t USING
    ( SELECT *
      FROM   cls_customers_scd
    MINUS
      SELECT *
      FROM   bl_dm.dim_customers_scd
    ) c ON ( c.customer_surr_id = t.customer_surr_id )
```

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 CUSTOMER_SURR_ID. Если совпадение есть, то происходит просто обновление флага в итоговой таблице IS_ACTIVE.

```
    WHEN matched THEN
    UPDATE SET t.is_active = c.is_active
    WHEN NOT matched THEN
    INSERT
```

- Если совпадения нет, то осуществляется INSERT.

```
WHEN NOT matched THEN
INSERT
  (
   customer_surr_id,
   customer_id,
   first_name,
   last_name,
   age,
   age_category,
   email,
   phone,
   address,
   city,
   country,
   region,
   start_dt,
   end_dt,
   is_active
  )
```

- Ну и конечно COMMIT.

```
  ) ;
  COMMIT;

EXCEPTION
  WHEN OTHERS THEN
  RAISE;
```

- Пример скрипта пакета.

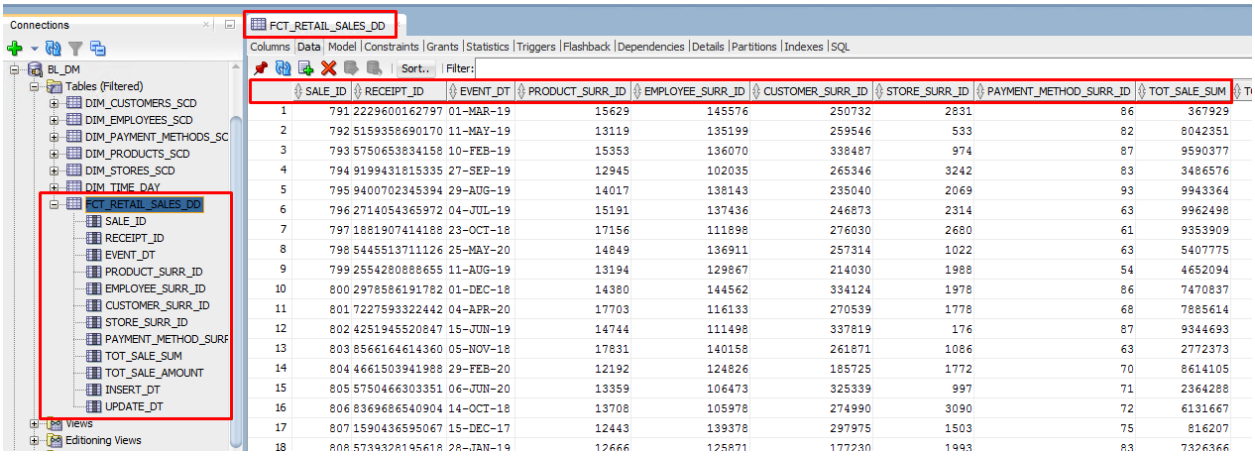
```
CREATE OR REPLACE PACKAGE pkg_etl_insert_customers
AUTHID CURRENT_USER
AS
  PROCEDURE insert_table_customers;
  PROCEDURE merge_table_customers;

END pkg_etl_insert_customers;
CREATE OR REPLACE PACKAGE BODY pkg_etl_insert_customers
AS
------------------------------------------------
PROCEDURE insert_table_customers…
END insert_table_customers;
------------------------------------------------
PROCEDURE merge_table_customers…
END merge_table_customers;
------------------------------------------------
END pkg_etl_insert_customers;
```

- Пакеты можно найти по директории **…\dwso\bl_cl_2nd\packages.**

4. Проверка информации на DM-слое

- Фактовая таблица **FCT_RETAIL_SALES_DD.**

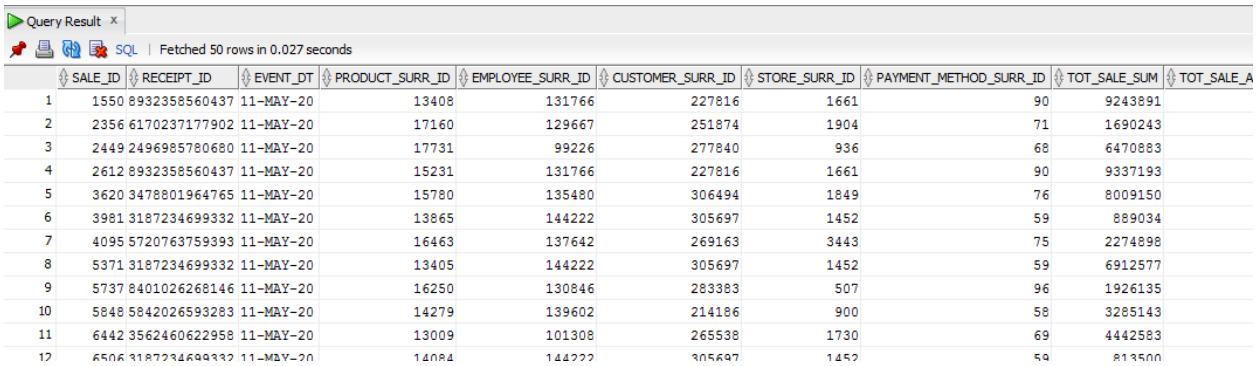


	SALE_ID	RECEIPT_ID	EVENT_DT	PRODUCT_SURR_ID	EMPLOYEE_SURR_ID	CUSTOMER_SURR_ID	STORE_SURR_ID	PAYMENT_METHOD_SURR_ID	TOT_SALE_SUM
1	791	2229600162797	01-MAR-19	15629	145576	250732	2831	86	367929
2	792	5159358690170	11-MAY-19	13119	135199	259546	533	82	8042351
3	793	5750653834158	10-FEB-19	15353	136070	338487	974	87	9590377
4	794	9199431815335	27-SEP-19	12945	102035	265346	3242	83	3486576
5	795	9400702345394	29-AUG-19	14017	138143	235040	2069	93	9943364
6	796	2714054365972	04-JUL-19	15191	137436	246873	2314	63	9962498
7	797	1881907414188	23-OCT-18	17156	111898	276030	2680	61	9353909
8	798	5445513711126	25-MAY-20	14849	136911	257314	1022	63	5407775
9	799	2554280888655	11-AUG-19	13194	129867	214030	1988	54	4652094
10	800	2978586191782	01-DEC-18	14380	144562	334124	1978	86	7470837
11	801	7227593322442	04-APR-20	17703	116133	270539	1778	68	7885614
12	802	4251945520847	15-JUN-19	14744	111498	337819	176	87	9344693
13	803	8566164614360	05-NOV-18	17831	140158	261871	1086	63	2772373
14	804	4661503941988	29-FEB-20	12192	124826	185725	1772	70	8614105
15	805	5750466303351	06-JUN-20	13359	106473	325339	997	71	2364288
16	806	8369686540904	14-OCT-18	13708	105978	274990	3090	72	6131667
17	807	1590436595067	15-DEC-17	12443	139378	297975	1503	75	816207
18	808	5739328195618	28-JAN-19	12666	125871	177230	1993	83	7326366

- Фактовая таблица **FCT_RETAIL_SALES_DD.**
- Проверка данных.

```
SELECT *
FROM fct_retail_sales_dd
WHERE event_dt = '11-MAY-20';
```

- Результат.

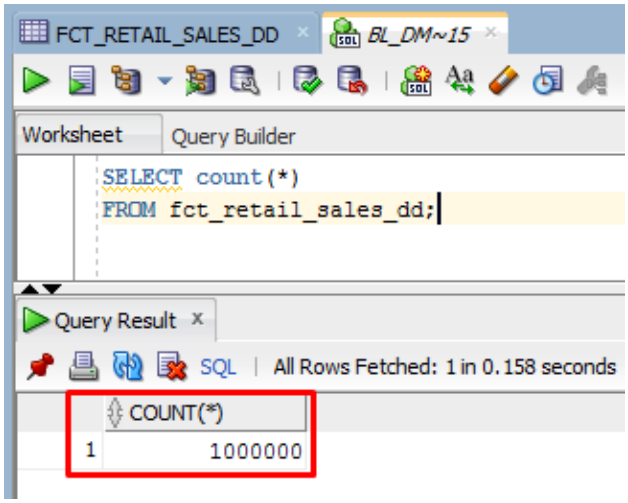


	SALE_ID	RECEIPT_ID	EVENT_DT	PRODUCT_SURR_ID	EMPLOYEE_SURR_ID	CUSTOMER_SURR_ID	STORE_SURR_ID	PAYMENT_METHOD_SURR_ID	TOT_SALE_SUM	TOT_SALE_A
1	1550	8932358560437	11-MAY-20	13408	131766	227816	1661	90	9243891	
2	2356	6170237177902	11-MAY-20	17160	129667	251874	1904	71	1690243	
3	2449	2496985780680	11-MAY-20	17731	99226	277840	936	68	6470883	
4	2612	8932358560437	11-MAY-20	15231	131766	227816	1661	90	9337193	
5	3620	3478801964765	11-MAY-20	15780	135480	306494	1849	76	8009150	
6	3981	3187234699332	11-MAY-20	13865	144222	305697	1452	59	889034	
7	4095	5720763759393	11-MAY-20	16463	137642	269163	3443	75	2274898	
8	5371	3187234699332	11-MAY-20	13405	144222	305697	1452	59	6912577	
9	5737	8401026268146	11-MAY-20	16250	130846	283383	507	96	1926135	
10	5848	5842026593283	11-MAY-20	14279	139602	214186	900	58	3285143	
11	6442	3562460622958	11-MAY-20	13009	101308	265538	1730	69	4442583	
12	6506	3187234699332	11-MAY-20	14084	144222	305697	1452	59	813500	

- Количество строк в таблице.

```
SELECT count(*)
FROM fct_retail_sales_dd;
```

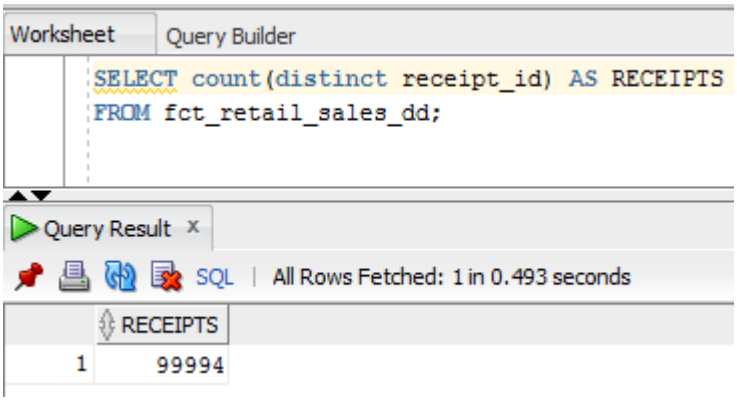
- Результат.



- Количество уникальных чеков.

```
SELECT count(distinct receipt_id) AS RECEIPTS
FROM fct_retail_sales_dd;
```

- Результат: чеков намного меньше, чем строк в таблице, поскольку один чек может включать несколько продуктов, а GRAIN таблицы именно продажи отдельно продуктов.



- Можно увидеть, что действительно в таблице миллион строк с продуктами. Здесь не применялся DISTINCT, поскольку один и тот же продукт мог продаваться в разные дни.