

EPAM Systems, RD Dep.

Overview Extraction

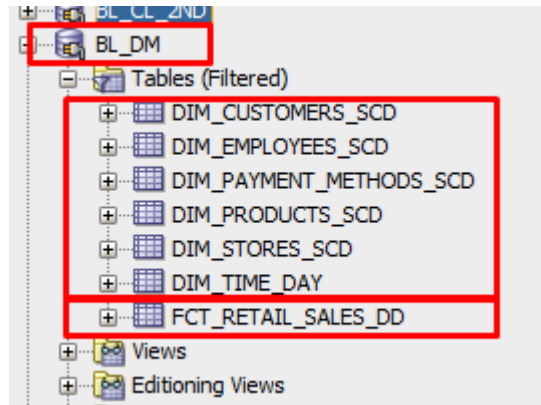
REVISION HISTORY					
Ver.	Description of Change	Author	Date	Approved	
				Name	Effective Date
1.0	Initial status	Valeryia Lupanova	30-NOV-2017		

Содержание

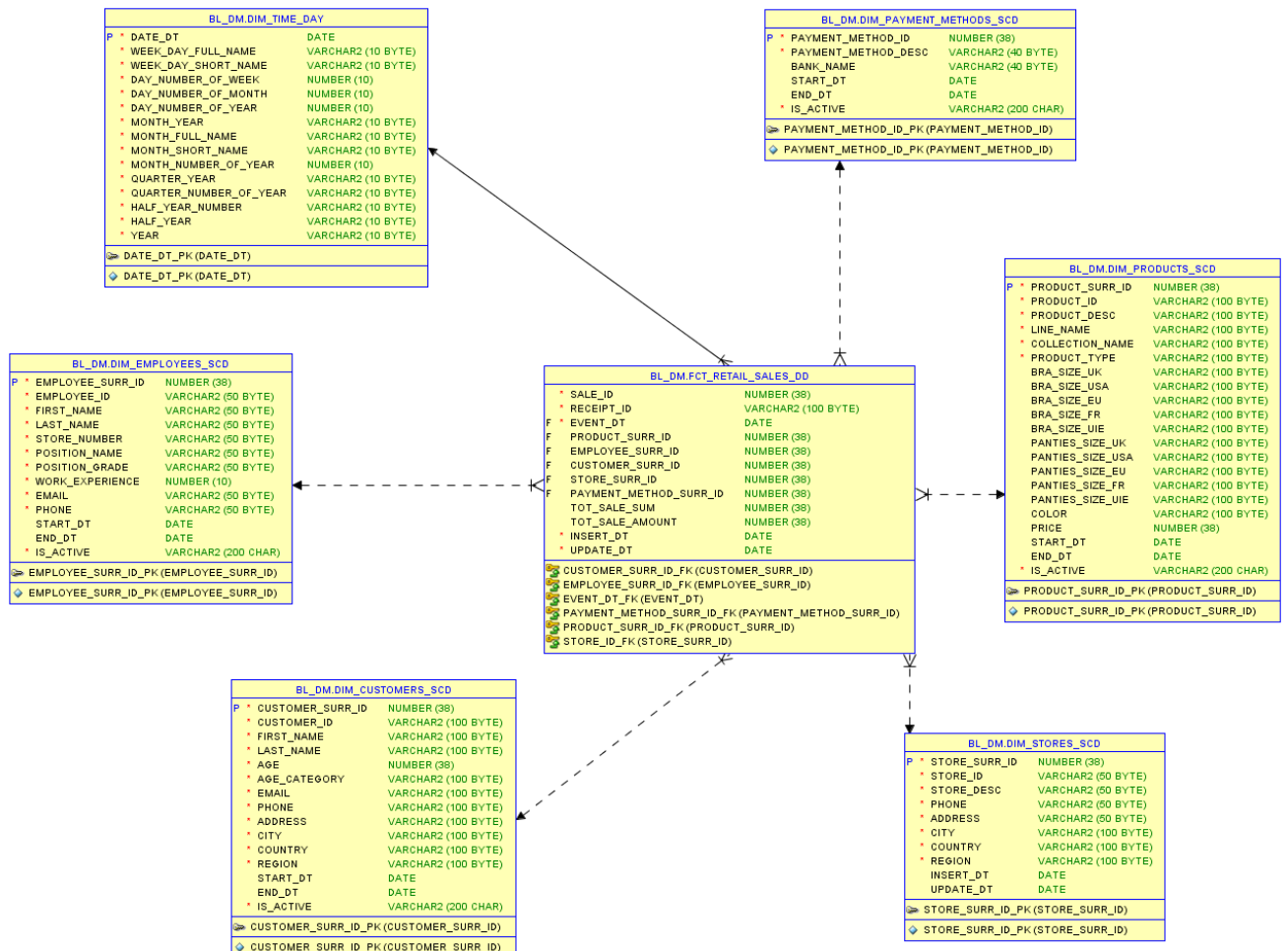
1. СОЗДАНИЕ FAST-TABLE.....	3
2. ВЫДАЧА ГРАНТОВ CL-СЛОЮ	9
3. СОЗДАНИЕ ПАКЕТОВ ДЛЯ ЗАПОЛНЕНИЯ FAST-ТАБЛИЦЫ.....	11
4. ПРОВЕРКА ИНФОРМАЦИИ НА DM-СЛОЕ	13

1. Создание FACT-table

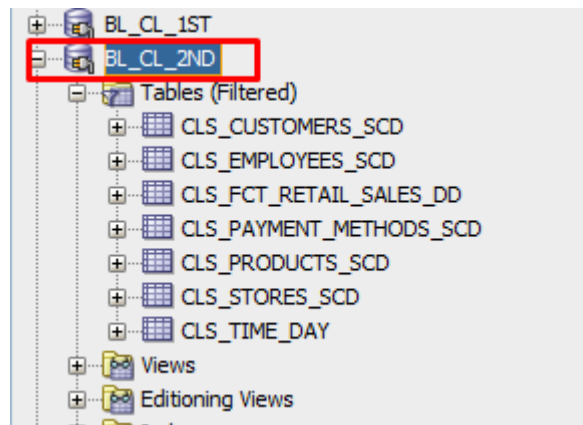
- На DM-слое была создана одна фактовая таблица – TRANSACTION.



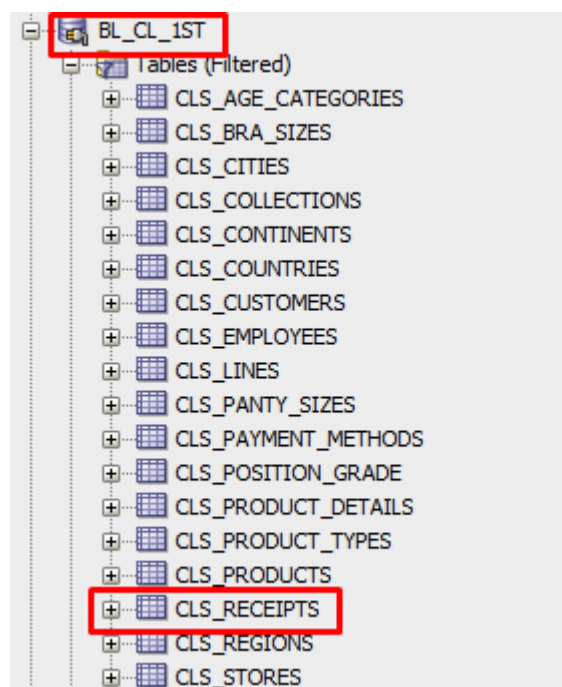
- Схема DM-слоя. Для DM-слоя, как видно, была выбрана схема STAR. Фактовая таблица содержит FOREIGN KEYS на другие таблицы слоя, однако, своего PRIMARY KEY нет.



- Следующим шагом была создана таблица на втором клинзинговом слое BL_CL_2ND, повторяющие структуру фактовой таблицы.



- Данные для фактовой таблицы генерировались на BL_CL_1ST-слое на основе данных, загруженных в 3NF-таблицы. Генерация данных была произведена в последнюю очередь, после загрузки всех дименшенов. На BL_CL_1ST-слое расположена CLS-таблица, служащая фреймворком для сгенерированных данных.



- Скрипт на генерацию данных.

```
CREATE OR REPLACE PACKAGE BODY pkg_etl_insert_receipts
AS
```

```

PROCEDURE insert_table_receipts
IS
BEGIN
EXECUTE IMMEDIATE ('TRUNCATE TABLE cls_receipts');
BEGIN
INSERT INTO cls_receipts (
.....
)
SELECT .....
FROM (
SELECT TRUNC(dbms_random.value(100000000000, 999999999999))
AS receipt_id ,
TRUNC ( (sysdate + 4) + dbms_random.value ( 1, 1000 )
) AS receipt_dt ,
ROUND ( dbms_random.value (
( SELECT MIN ( store_id ) FROM bl_3nf.ce_stores),
( SELECT MAX ( store_id ) FROM bl_3nf.ce_stores) ) )
AS store_id ,
ROUND ( dbms_random.value (
( SELECT MIN ( employee_id)
FROM bl_3nf.ce_employees
WHERE UPPER(SUBSTR(TRIM(is_active),1,4))='TRUE' ),
.....
FROM (SELECT * FROM dual connect by level <1000000)
);
END;
COMMIT;
EXCEPTION
WHEN OTHERS THEN
RAISE;
END insert_table_receipts;
-----

```

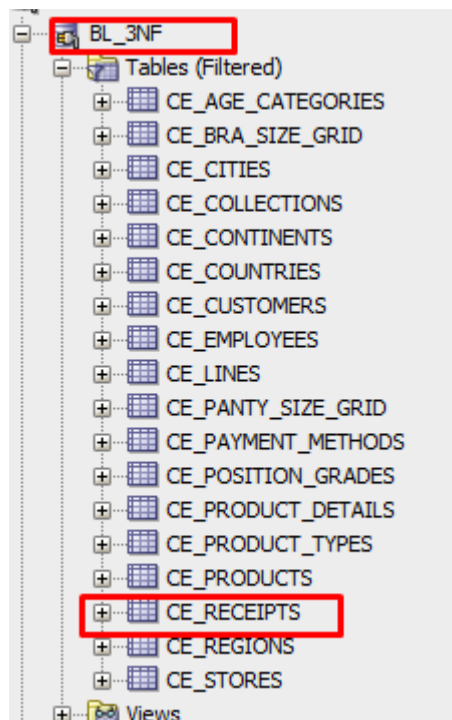
- Все данные генерировались с помощью **DBMS_RANDOM.VALUE**. В качестве граничных значений выбирались **MIN** и **MAX** по полю с идентификатором. **CONNECT BY LEVEL <1000000** – позволяет создать миллион строк. Также в операторах можно наблюдать условие, по которому фильтруются случайные данные. Все объекты, имеющие флаг, должны быть в активном статусе. Генерация данных вызывается процедурой из пакета:

```

BEGIN
pkg_etl_insert_receipts.insert_table_receipts;
END;

```

- После генерации данные загружаются в таблицу на 3NF-слое. В качестве **CONSTRAINTS** выступают таблицы, на базе которых происходила генерация данных. Внешние ключи называются на 3NF-слое как **ID** и являются суррогатными для 3NF-слоя.



- Заполнение таблицы CE_RECEIPTS (3NF-слой) осуществляется с помощью MERGE.

```

-----
PROCEDURE merge_table_ce_receipts
IS
BEGIN
MERGE INTO bl_3nf.ce_receipts t USING
    ( SELECT ...
      MINUS
        SELECT ...
        FROM   bl_3nf.ce_receipts
      ) c ON (
          c.receipt_id = t.receipt_id
        AND
          c.insert_dt = t.insert_dt
        AND
          c.receipt_dt = t.receipt_dt
        AND
          c.customer_id = t.customer_id
        AND
          c.employee_id = t.employee_id
        AND
          c.receipt_sum = t.receipt_sum_usd
        AND
          c.product_detail_id = t.product_detail_id
      )
  WHEN NOT matched THEN
  INSERT
  (
    ...
  )
  VALUES
  (
    ...
  ) ;
COMMIT;

```

```

EXCEPTION
WHEN OTHERS THEN
    RAISE;
END merge_table_ce_receipts;
-----

```

- При загрузке в 3NF-проверяются все атрибуты, поскольку абсолютно идентичных чеков быть не может.
- Далее данные перезагружаем. При перезагрузке данных в CLS второго слоя делаем JOINS по полям с внешними ключами на CE-таблицы, чтобы данные по ключевым полям точно существовали в БД.

```

-----
PROCEDURE insert_table_retail_sales
IS
BEGIN
    EXECUTE IMMEDIATE ('TRUNCATE TABLE cls_fct_retail_sales_dd');
    INSERT INTO cls_fct_retail_sales_dd
    SELECT receipt_id,
           cr.receipt_dt AS event_dt,
           ce.employee_id AS employee_surr_id,
           cm.customer_id AS customer_surr_id,
           ct.store_id AS store_id,
           cr.payment_method_id AS payment_method_surr_id,
           cr.product_detail_id AS product_detail_surr_id,
           cr.receipt_sum_usd AS tot_sale_sum,
           ROUND ( dbms_random.value( 100, 99999), 2) AS tot_sale_amount,
           cr.insert_dt AS insert_dt,
           SYSDATE AS update_dt
    FROM    bl_3nf.ce_receipts cr left join bl_3nf.ce_employees ce
           on cr.employee_id = ce.employee_id
           left join bl_3nf.ce_customers cm
           on cr.customer_id = cm.customer_id
           left join bl_3nf.ce_stores ct
           on cr.store_id = ct.store_id
           left join bl_3nf.ce_product_details cpd
           on cr.product_detail_id =
cpd.product_details_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        RAISE;
END insert_table_retail_sales;
-----

```

- Последним шагом данные перезагружаем в FCT-таблицу. Поскольку наша модель не требует дополнительных агрегаций, то никаких модификаций с данными происходить не будет. При перезагрузке данных формируем SEQUENCE на стороне DM-слоя для уникального поля SALE_ID в фактовой таблице. Также производим JOINS нашей CLS-таблицы с дименшенами, для того чтобы присвоить внешним ключам новые значения.

Вместо ID – суррогатного ключа 3NF-слоя – присваиваем на базе ID значение SURR_ID – суррогатный ключ DM-слоя. Для этого в процедуре загрузки в JOIN операции указываем ID поле, но вытягиваем из DM-слоя аналог – SURR_ID.

```

-----
PROCEDURE insert_table_fct_retail_sales
IS
BEGIN
    INSERT INTO bl_dm.fct_retail_sales_dd
    SELECT bl_dm.fct_retail_sales_dd_seq.NEXTVAL AS sale_id,
           a.receipt_id,
           a.event_dt,
           c.employee_surr_id,
           b.customer_surr_id,
           d.store_surr_id,
           d.payment_method_id AS payment_method_surr_id,
           e.product_surr_id,
           a.tot_sale_sum,
           a.tot_sale_amount,
           a.insert_dt,
           a.update_dt
    FROM   cls_fct_retail_sales_dd a LEFT JOIN
           bl_dm.dim_customers_scd b ON
           a.customer_surr_id = b.customer_id
           LEFT JOIN
           bl_dm.dim_employees_scd c ON
           a.employee_surr_id = c.employee_id
           LEFT JOIN
           bl_dm.dim_stores_scd d ON
           a.store_surr_id = d.store_id
           LEFT JOIN
           bl_dm.dim_payment_methods_scd d ON
           a.payment_method_surr_id = d.payment_method_id
           LEFT JOIN
           bl_dm.dim_products_scd e ON
           a.product_surr_id = e.product_id;
    COMMIT;
EXCEPTION
WHEN OTHERS THEN
    RAISE;
END insert_table_fct_retail_sales;
-----

```


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.

```

BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'SELECT', SCHEMA_NAME => 'BL_3NF', OBJECT_NAME => 'CE_CONTINENTS', USER_NAME => 'BL_CL_2ND');
END;

BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'SELECT', SCHEMA_NAME => 'BL_3NF', OBJECT_NAME => 'CE_REGIONS', USER_NAME => 'BL_CL_2ND');
END;

BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'SELECT', SCHEMA_NAME => 'BL_3NF', OBJECT_NAME => 'CE_COUNTRIES', USER_NAME => 'BL_CL_2ND');
END;

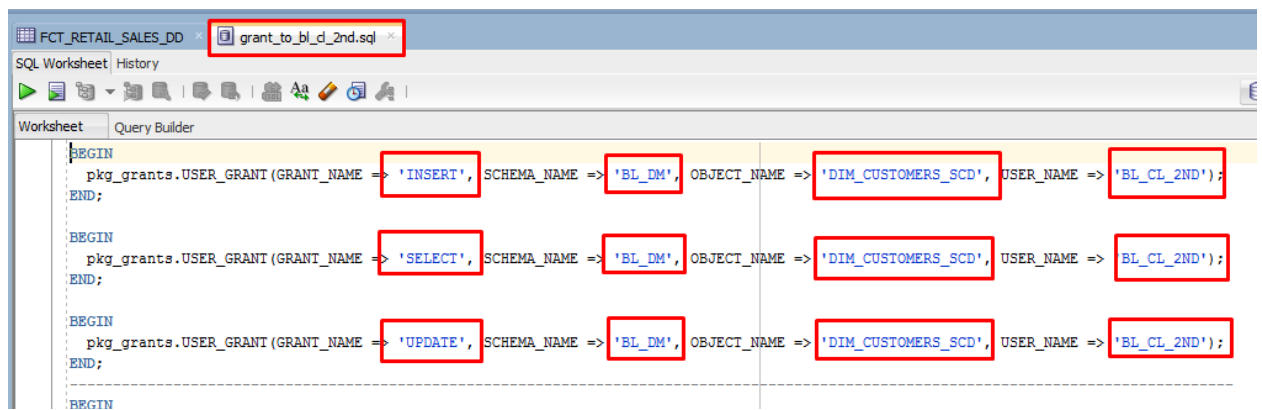
BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'SELECT', SCHEMA_NAME => 'BL_3NF', OBJECT_NAME => 'CE_CITIES', USER_NAME => 'BL_CL_2ND');
END;
  
```

- Файл с грантами расположен в папке BL_3NF.

DATA (D:) > Valeria_Lupanova > Project > dwso > bl_3nf > grants				
Name	Date modified	Type	Size	
grant_to_bl_cl_1st.sql	11/25/2017 7:18 PM	Microsoft SQL Ser...	14 KB	
grant_to_bl_cl_2nd.sql	11/26/2017 3:54 PM	Microsoft SQL Ser...	6 KB	

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

> DATA (D:) > Valeria_Lupanova > Project > dwso > bl_dm > grants				
Name	Date modified	Type	Size	
grant_to_bl_cl_2nd.sql	11/26/2017 5:57 PM	Microsoft SQL Ser...	5 KB	



```
BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'INSERT', SCHEMA_NAME => 'BL_DM', OBJECT_NAME => 'DIM_CUSTOMERS_SCD', USER_NAME => 'BL_CL_2ND');
END;

BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'SELECT', SCHEMA_NAME => 'BL_DM', OBJECT_NAME => 'DIM_CUSTOMERS_SCD', USER_NAME => 'BL_CL_2ND');
END;

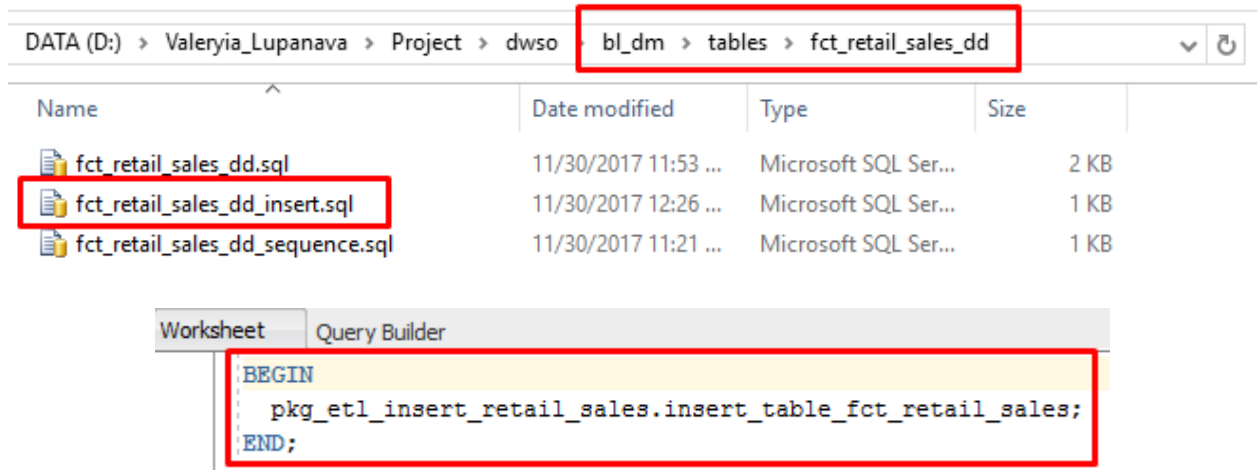
BEGIN
  pkg_grants.USER_GRANT (GRANT_NAME => 'UPDATE', SCHEMA_NAME => 'BL_DM', OBJECT_NAME => 'DIM_CUSTOMERS_SCD', USER_NAME => 'BL_CL_2ND');
END;

BEGIN
```

- Было использовано четыре типа грантов: INSERT, UPDATE, два SELECT.
 - INSERT dm-table – при заполнении таблицы данными выполняется INSERT;
 - SELECT dm-table – при merge-операции необходимо сначала сделать выборку данных из dm-table для проверки на дубли перед заполнением;
 - UPDATE dm-table – при merge-операции, если попадает значение, которое ранее было добавлено, осуществляется UPDATE;
 - SELECT dm-sequence – для возможности генерации суррогатного ключа из BC_CL_2ND.

3. Создание пакетов для заполнения FACT-таблицы

- 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 фактовой таблицы использовалась операция INSERT, поскольку данные генерируются постоянно и вероятность создания дубля очень мала.
- Пример скрипта.

```

CREATE OR REPLACE PACKAGE pkg_etl_insert_retail_sales
AUTHID CURRENT_USER
AS
  PROCEDURE insert_table_retail_sales;
  PROCEDURE insert_table_fct_retail_sales;
END pkg_etl_insert_retail_sales;
CREATE OR REPLACE PACKAGE BODY pkg_etl_insert_retail_sales
AS
-----
PROCEDURE insert_table_retail_sales
IS
BEGIN
  .....
END insert_table_retail_sales;
-----
PROCEDURE insert_table_fct_retail_sales
IS
BEGIN
  .....
EXCEPTION
WHEN OTHERS THEN
  RAISE;
END insert_table_fct_retail_sales;
-----
END pkg_etl_insert_retail_sales;

```

- При использовании INSERT данные будут постоянно дописываться в таблицу фактов.
- Пакеты можно найти по директории ...\\dwso\\bl_cl_2nd\\packages.

4. Проверка информации на DM-слое

- Фактовая таблица **FCT_RETAIL_SALES_DD**.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables (Filtered)' list includes 'FCT_RETAIL_SALES_DD'. The main pane displays the table's columns and a sample of data rows. The columns are: 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, and TOT_SALE_AMOUNT. The data rows show various sales transactions with their respective IDs, dates, and sums.

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_AMOUNT
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	1443	83	7326366	

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

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

- Результат.

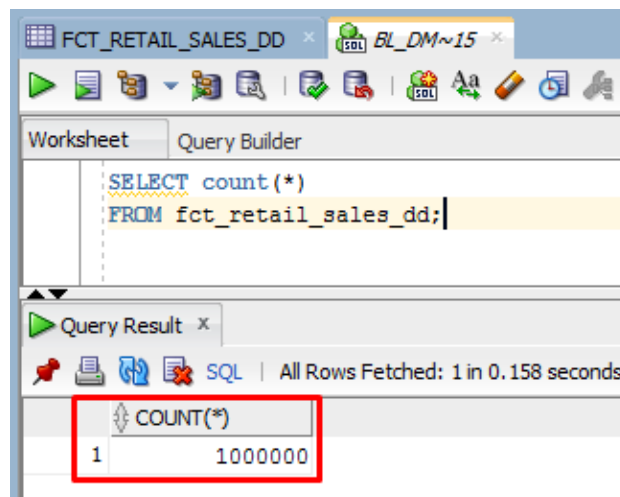
The screenshot shows the 'Query Results' window in SQL Server. It displays 12 rows of data for the date '11-MAY-20'. The columns are the same as in the previous table. The data shows various sales transactions with their respective IDs, dates, and sums.

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_AMOUNT
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;
```

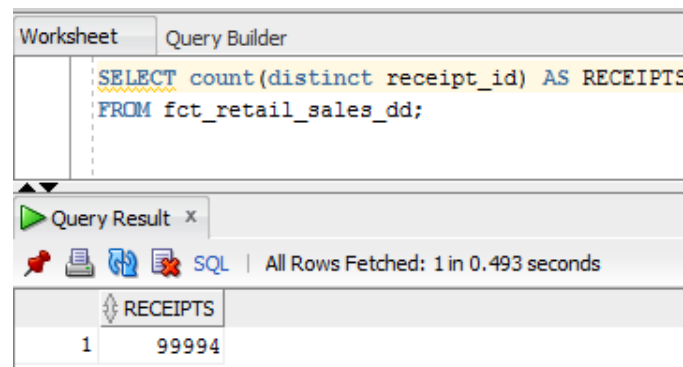
- Результат.



- 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.

