Sales analysis

MILAVITSA'S SALES ANALYSIS

Milavitsa-Clothes

CONTENTS

1 BUSINESS DESCRIPTION 4
1.1 BUSINESS BACKGROUND 4
1.2 PROBLEMS BECAUSE OF POOR DATA MANAGEMENT 4
1.3 BENEFITS FROM IMPLEMENTING A DATA WAREHOUSE 5
2 DATA DESCRIPTION 6
2.1 DESCRIPTION OF SELECTED SCHEMA 6
2.2 DESCRIPTION OF DATA SOURCES 6
2.3 DESCRIPTION OF BUSINESS RULES 6
3 DIMENSIONS OF A BUSINESS 7
3.1 DEFINITION OF SELECTED BUSINESS PROCESS 7
3.2 DECLARING THE GRAIN 7
3.3 IDENTIFYING THE DIMENSIONS 7
3.4 IDENTIFYING THE FACTS 7
3.5 DIMENSIONAL MODEL. STAR SCHEMA 8
4 NF-LAYER OF DATA WAREHOUSE 9
4.1 3NF-MODEL 13
5 OBJECT PARTITIONING 17
6 BUSINESS PROCESSES 19
6.1 SALES ANALYSIS 20
6.2 ANALYSIS OF RECEIPTS 21
6.3 LOAD ANALYSIS 22
7 DATA MODELLING 23
7.1 DETAILING DIAGRAMS FOR 3NF AND STAR/SNOWFLAKE LAYERS 23
7.2 TEXTUAL DESCRIPTION OF LAYERS OF DATA WAREHOUSE 23
7.2.1 Source layer 23
7.2.2 Data staging layer 23
7.2.3 Data warehouse layer 24
7.2.4 Analysis layer 24
7.3 VISUAL DESCRIPTION OF LAYERS OF DATA WAREHOUSE 25
8 FACT TABLE PARTITIONING STRATEGY 31
9 STRATEGY OF PARALLEL LOAD 32

10 REPORT LAYOUTS 37
10.1 3NF-LAYER 37
10.2 DM-LAYER 40

1 BUSINESS DESCRIPTION

1.1 BUSINESS BACKGROUND

For many years Milavitsa has been producing lady's lingerie, being one of the biggest lingerie producer in Eastern Europe. The company's products are successfully sold in more than 25 countries around the world and are characterized by high quality, original and fashionable designs.

Company has four main brands:

1. Milavitsa.

The Milavitsa product portfolio includes bras, knickers, shapewear, knitwear and swimwear. The Milavitsa collection is divided into three categories: classic, fashion and swimwear. The basis of the classic collection is formed by a large variety of everyday styles, combining the basics of classical design, comfortable construction and functional materials. Milavitsa is an expert in creating the styles of large sizes. Fashion and swimwear collections are created for every season, following the fashion trends in design, materials and accessories.

2. Avelive Collection.

Bras, knickers, shapewear are also manufactured under the Aveline trademark. This is comfortable everyday lingerie at affordable prices. The Aveline products are developed and manufactured by Milavitsa, which is a guarantee of high quality and compliance with the company's standards.

3. Alisee.

Alisee is a French lingerie brand acquired by Milavitsa. Alisee collection is designed and styled by European professionals. Tailored by Milavitsa to fit local market specifics.

4. Hidalgo.

The Hidalgo is men's underwear combines classic shapes and comfortable natural materials.

1.2 PROBLEMS BECAUSE OF POOR DATA MANAGEMENT

Company faced next problems:

1. absence of business intelligence from several sources;
2. absence of sales and inventory information consolidation for the calculation of the optimal order and delivery;
3. decreasing query and system performance;
4. absence of timely access to data;
5. absence of historical intelligence.

1.3 BENEFITS FROM IMPLEMENTING A DATA WAREHOUSE

1. Better decision-making.

   Corporate decision makers will no longer have to make important business decisions based on limited data and hunches. Data warehouse will store credible facts and statistics, and decision makers will be able to retrieve that information from the data warehouse based on their personal needs.

2. Quick and easy access to data.

   Speed is an important factor that sets company above its competitors. Business users can quickly access data from multiple sources from a data warehouse, meaning that precious time will not be wasted on retrieving data from multiple sources. This allows company to make quick and accurate decisions, with little or no support from its IT department.

3. Data quality and consistency.

   Since data warehouses gather information from different sources and convert it into a single and widely used format, departments will produce results that are in line and consistent with each other. When data is standardized, company can have confidence in its accuracy, and accurate data is what makes for strong business decisions.

2 DATA DESCRIPTION

2.1 DESCRIPTION OF SELECTED SCHEMA

The Star schema was chosen for business processes description.

The main reasons:

1. simple structure;
2. absence of a big number of tables to join;
3. denormalized tables are not too large in a specific case of this task;
4. widely support by a large number of business intelligence tools.

2.2 DESCRIPTION OF DATA SOURCES

Customer and manager information was generated on the site www.mockaroo.com. It allows creating file with 1000 rows.

Additionally, some specific information was modified in the Excel. Information in these columns was created by the specific random formula.

Information for Collection, Line, Product type, Store dimensions is on the official company's site www.milavitsa.com. Information about Sizes can be find via tne next links www.milavitsa.com/collections/converter and www.globebrand.com/sizing_charts

2.3 DESCRIPTION OF BUSINESS RULES

Business rules:

1. Every retail sale must be associated with a valid Customer.
2. A retail sale is always associated with a Payment Method.
3. A retail sale can have one or many Products.
4. A retail sale is always associated with a specific Employee.

3 DIMENSIONS OF A BUSINESS

3.1 DEFINITION OF SELECTED BUSINESS PROCESS

Business Process for analysis is Milavitsa's sales per different metrics.

3.2 DECLARING THE GRAIN

The highest fact granularity is daily sales amount per certain customer and employee in a specific a store.

3.3 IDENTIFYING THE DIMENSIONS

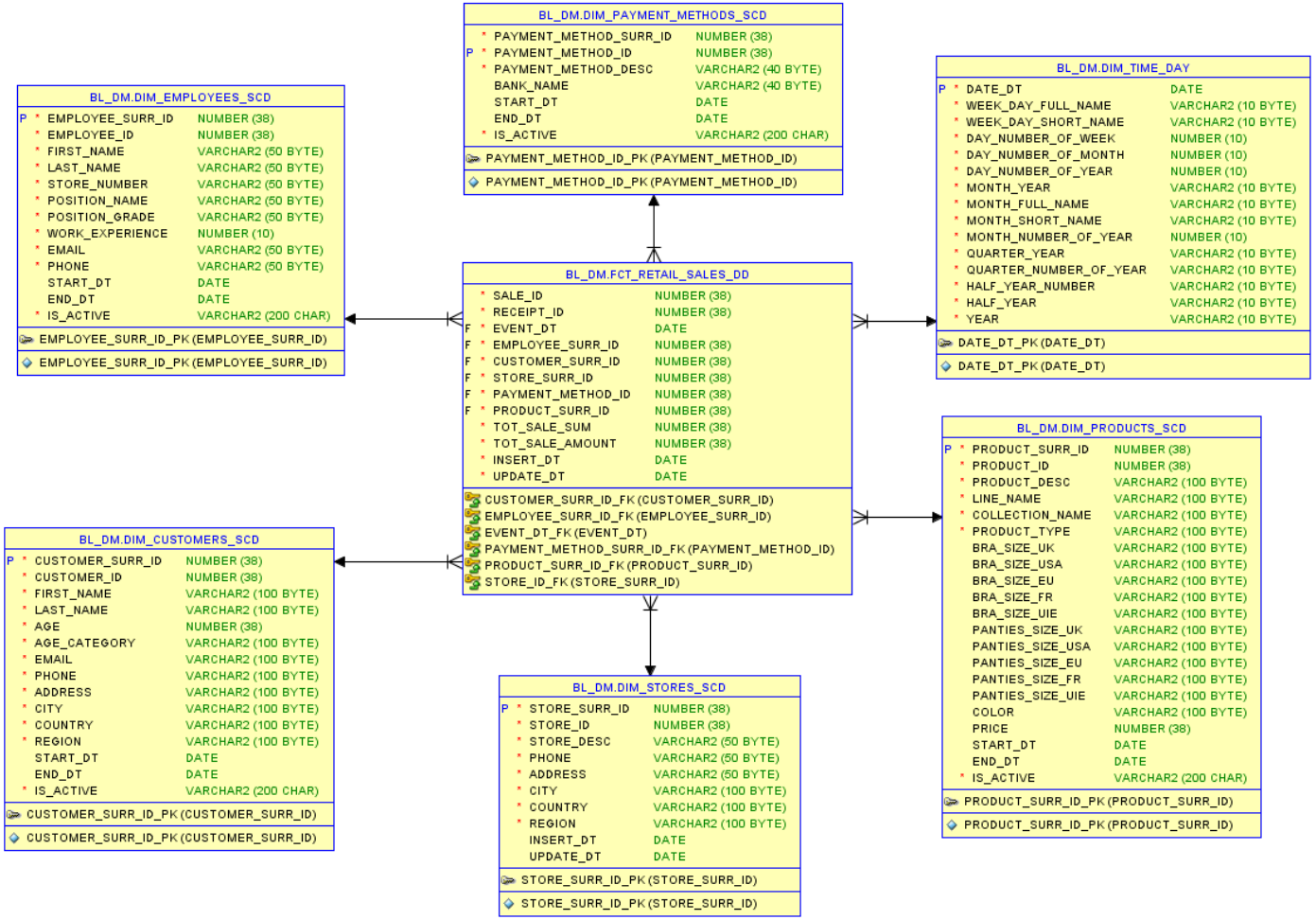
Schema should contains next dimensions:

1. Dim_customers_scd:
2. Dim_products_scd:
3. Dim_stores:
4. Dim_payment_methods_scd;
5. Dim_employees_scd;
6. Dim_time_day.

3.4 IDENTIFYING THE FACTS

Schema should contains the fct_retail_sales_dd fact table.

3.5 DIMENSIONAL MODEL. STAR SCHEMA

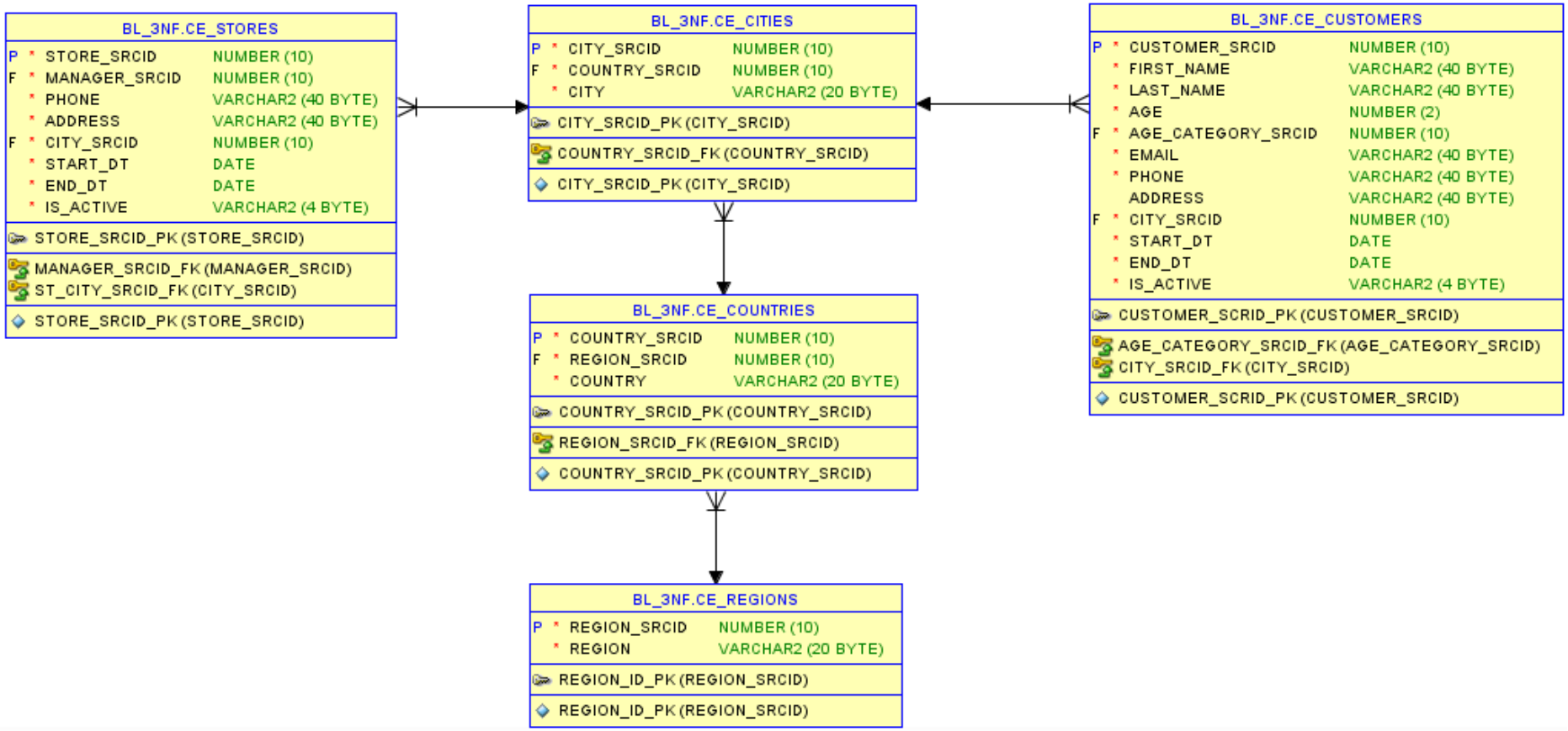


Picture 1 Dimensional model

4 NF-LAYER OF DATA WAREHOUSE

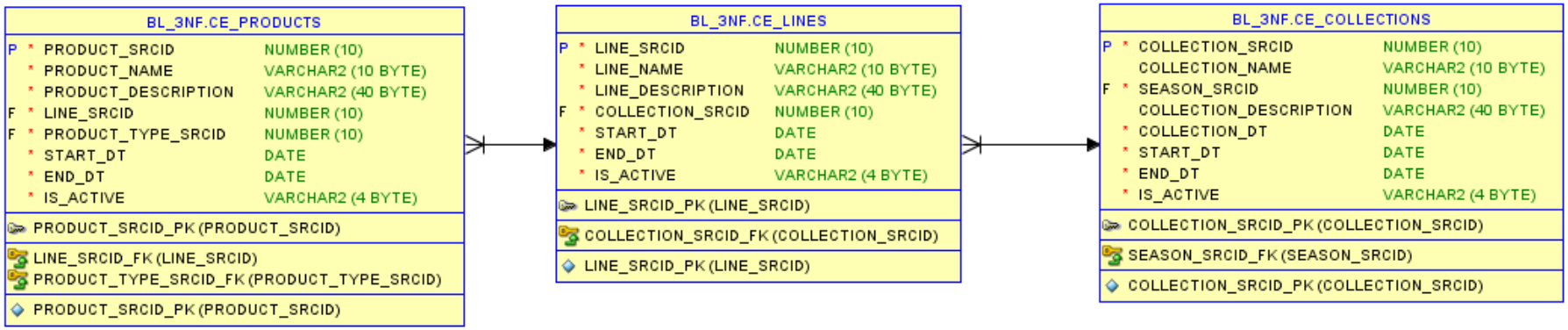
In this model the normalization was made with next steps:

1. Geographical information from the “Stores” and “Customers” was separated on different tables. Tables were created for each geographical object: region -> country -> city. These objects were connected in series: from the lowest in granularity to the highest.



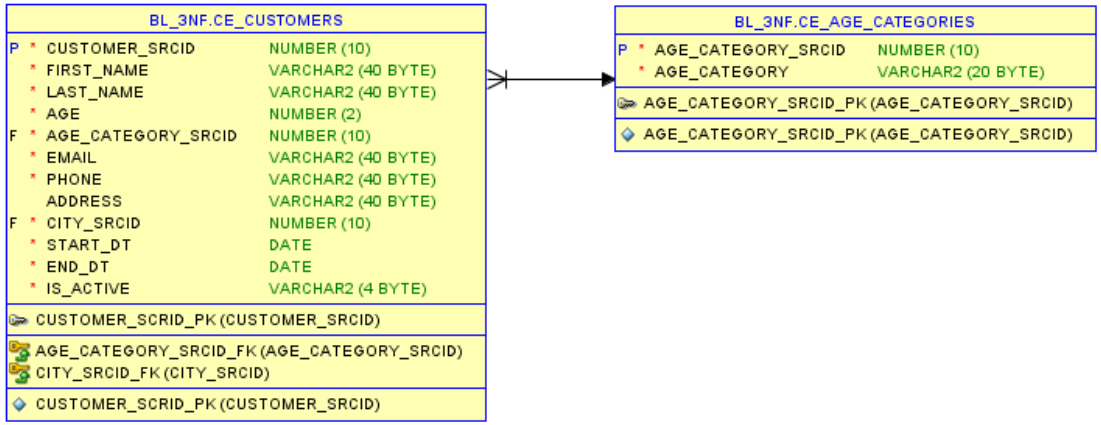
Picture 2 Geography part

2. Information about products was organized in the same way. There were separate tables for each object: collections -> lines -> products. These objects were connected in series: from the lowest in granularity to the highest.



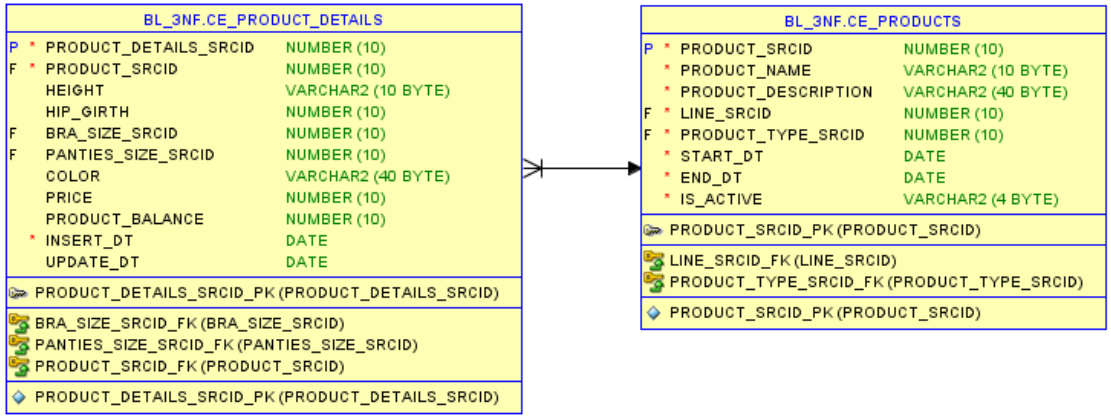
Picture 3 Product hierarchy part

3. A separate “Age_categories” table was created for Customers. This table describes all possible age ranges that are important for business. Subsequently, each Customer can be assigned to a specific age category.



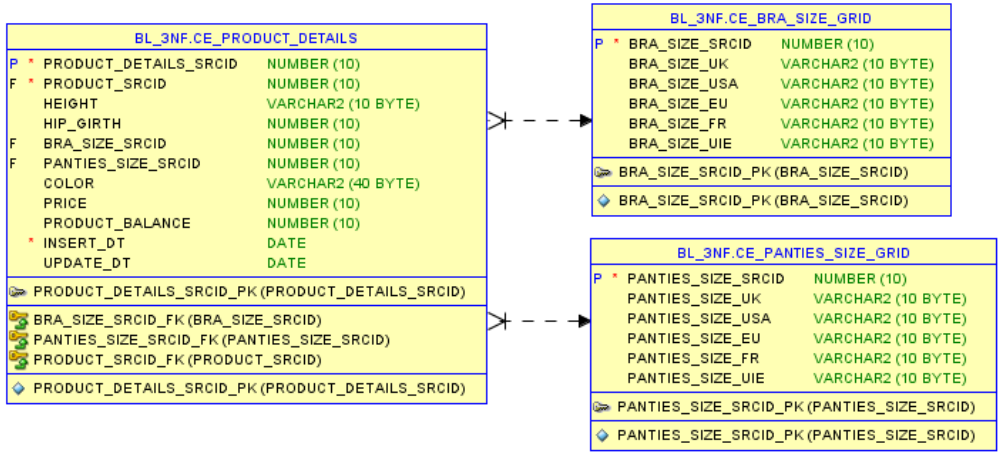
Picture 4 Customer part

4. Information about product characteristics was separated in a specific table “Product_details”. It contains a complete description of each commodity unit of a certain size.



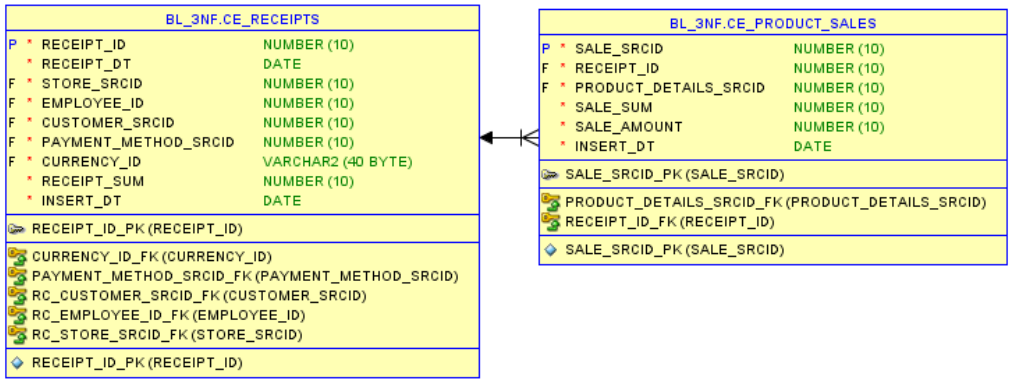
Picture 5 Product part

5. The tables of the size grids “Bra_size_grid” and “Panities_size_grid”, which contain information on the international dimension matching for bra and penities, were separately created.



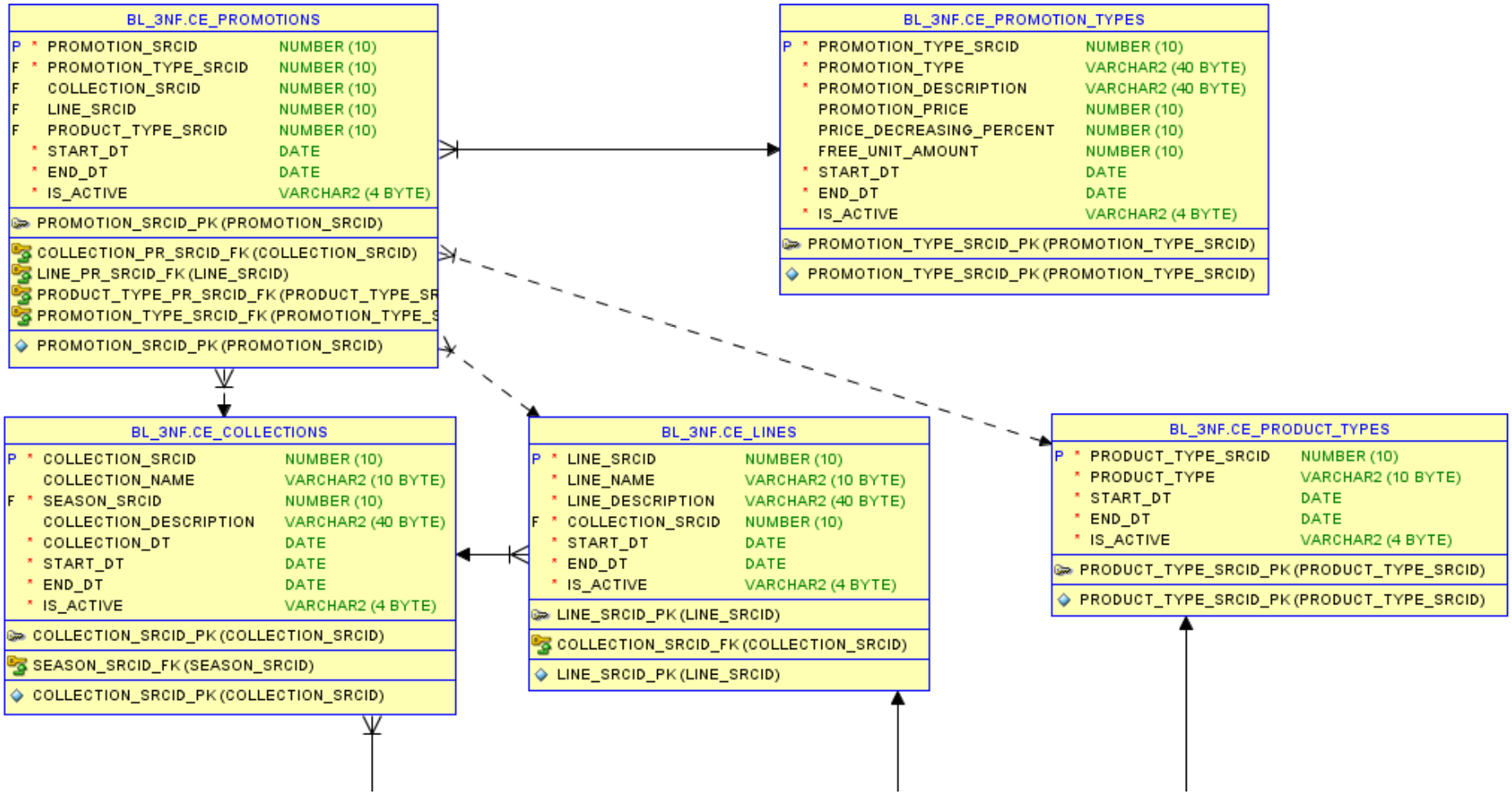
Picture 6 Size grid part

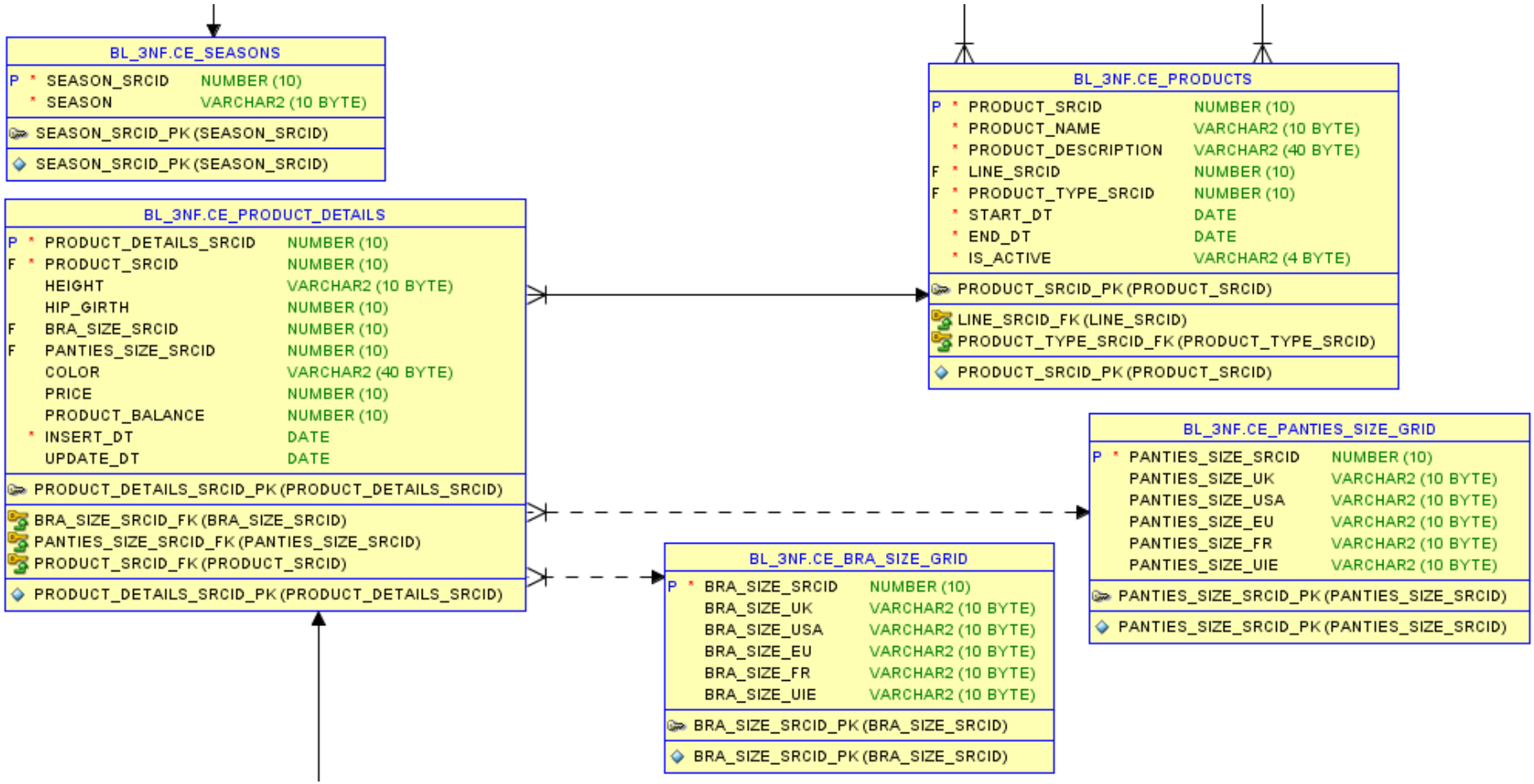
6. Additionally receipts information was add. It includes the calculation of their total cost.
7. The “Product_sales” table details the check information, namely it describes a concrete product and amount of units what were purchased.

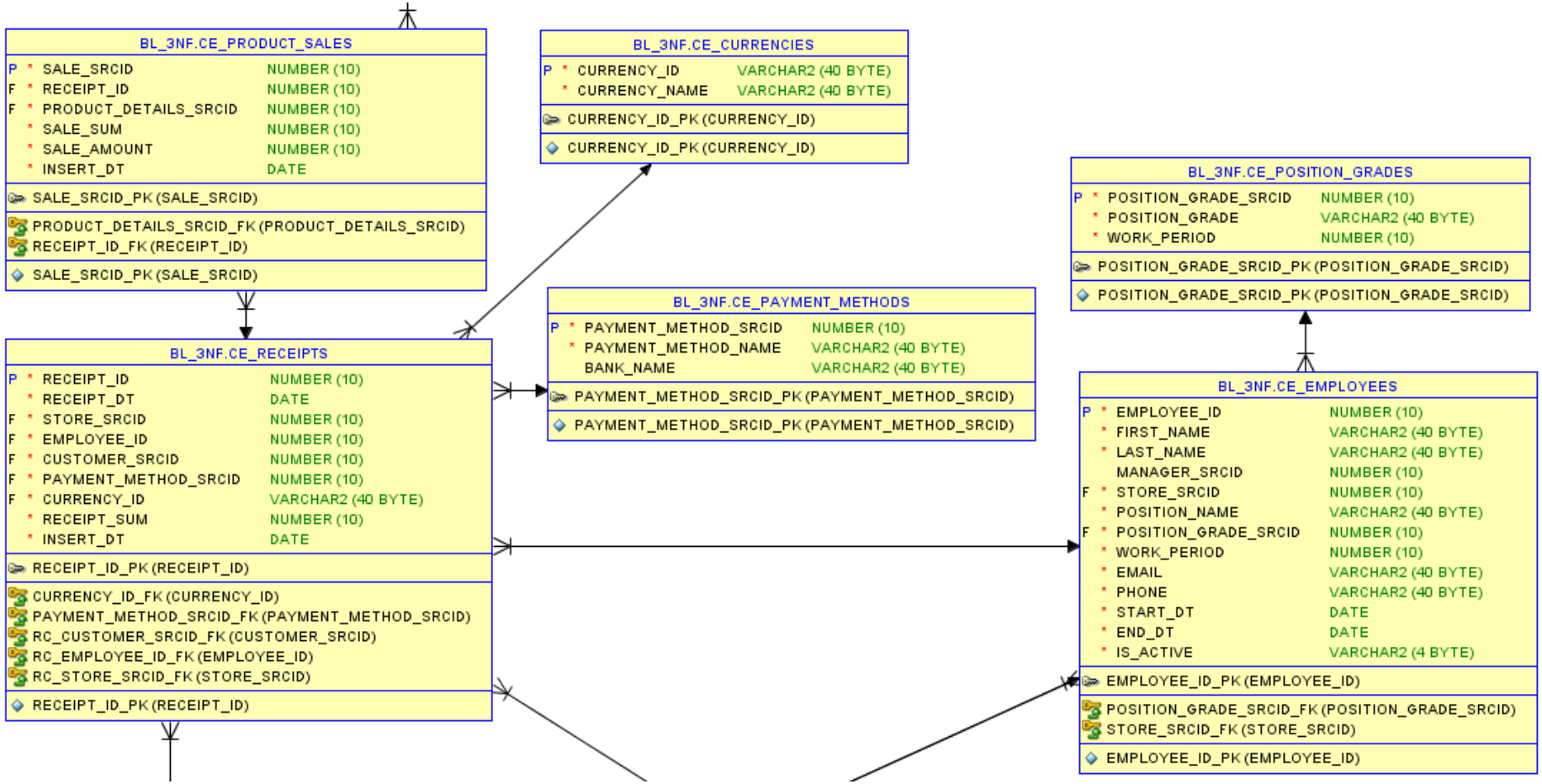


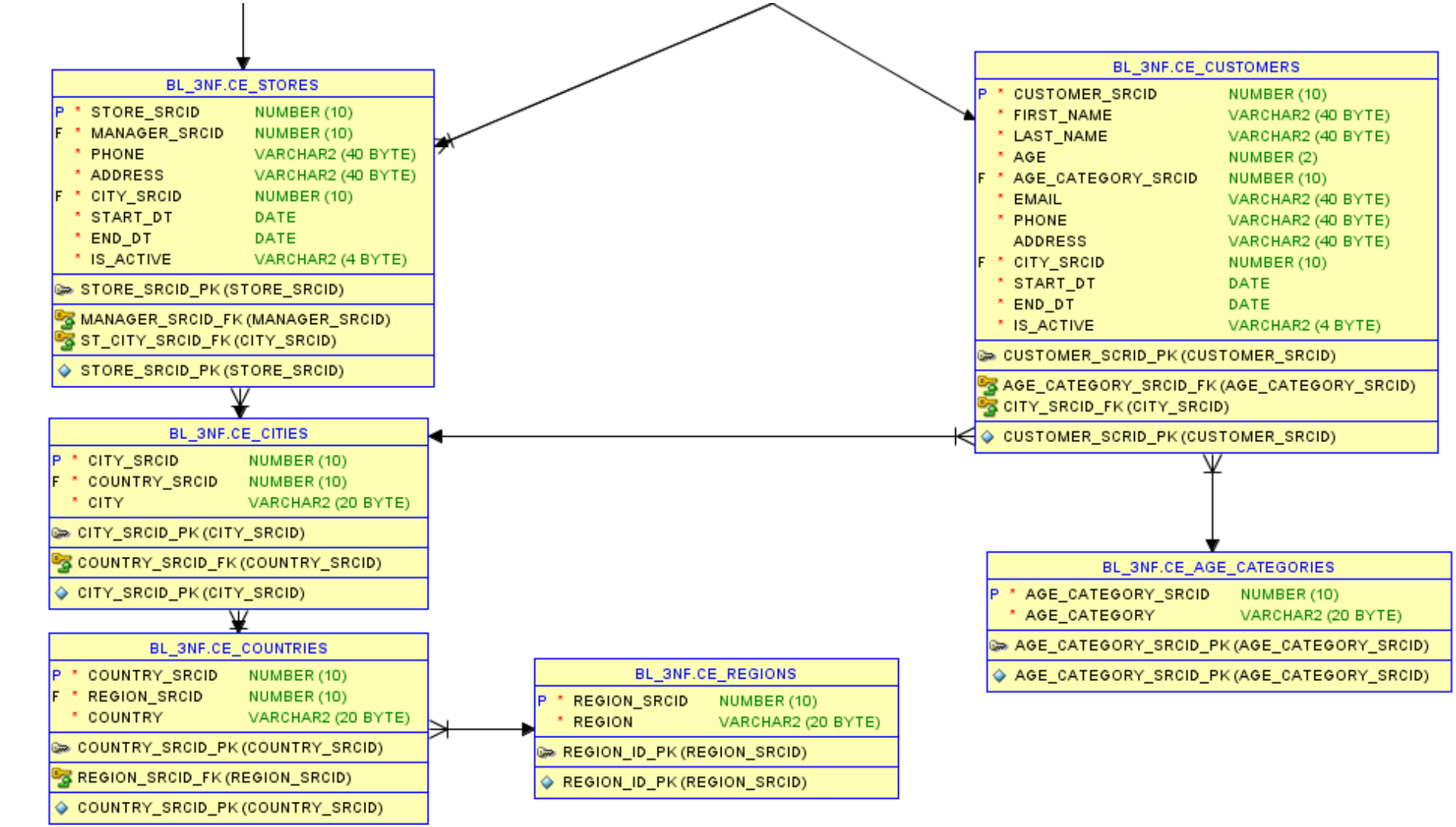
Picture 7 Sales part

4.1 3NF-MODEL


BL_3NF.CE_PROMOTIONS
P * PROMOTION_SRCID NUMBER (10)
F * PROMOTION_TYPE_SRCID NUMBER (10)
F COLLECTION_SRCID NUMBER (10)
F LINE_SRCID NUMBER (10)
F PRODUCT_TYPE_SRCID NUMBER (10)
* START_DT DATE
* END_DT DATE
* IS_ACTIVE VARCHAR2 (4 BYTE)
PROMOTION_SRCID_PK (PROMOTION_SRCID)
COLLECTION_PR_SRCID_FK (COLLECTION_SRCID)
LINE_PR_SRCID_FK (LINE_SRCID)
PRODUCT_TYPE_PR_SRCID_FK (PRODUCT_TYPE_SR
PROMOTION_TYPE_SRCID_FK (PROMOTION_TYPE_S
PROMOTION_SRCID_PK (PROMOTION_SRCID)
BL_3NF.CE_PROMOTION_TYPES
P * PROMOTION_TYPE_SRCID NUMBER (10)
* PROMOTION_TYPE VARCHAR2 (40 BYTE)
* PROMOTION_DESCRIPTION VARCHAR2 (40 BYTE)
PROMOTION_PRICE NUMBER (10)
PRICE_DECREASING_PERCENT NUMBER (10)
FREE_UNIT_AMOUNT NUMBER (10)
* START_DT DATE
* END_DT DATE
* IS_ACTIVE VARCHAR2 (4 BYTE)
PROMOTION_TYPE_SRCID_PK (PROMOTION_TYPE_SRCID)
PROMOTION_TYPE_SRCID_PK (PROMOTION_TYPE_SRCID)
BL_3NF.CE_COLLECTIONS
P * COLLECTION_SRCID NUMBER (10)
COLLECTION_NAME VARCHAR2 (10 BYTE)
F * SEASON_SRCID NUMBER (10)
COLLECTION_DESCRIPTION VARCHAR2 (40 BYTE)
* COLLECTION_DT DATE
* START_DT DATE
* END_DT DATE
* IS_ACTIVE VARCHAR2 (4 BYTE)
COLLECTION_SRCID_PK (COLLECTION_SRCID)
SEASON_SRCID_FK (SEASON_SRCID)
COLLECTION_SRCID_PK (COLLECTION_SRCID)
BL_3NF.CE_LINES
P * LINE_SRCID NUMBER (10)
* LINE_NAME VARCHAR2 (10 BYTE)
* LINE_DESCRIPTION VARCHAR2 (40 BYTE)
F * COLLECTION_SRCID NUMBER (10)
* START_DT DATE
* END_DT DATE
* IS_ACTIVE VARCHAR2 (4 BYTE)
LINE_SRCID_PK (LINE_SRCID)
COLLECTION_SRCID_FK (COLLECTION_SRCID)
LINE_SRCID_PK (LINE_SRCID)
BL_3NF.CE_PRODUCT_TYPES
P * PRODUCT_TYPE_SRCID NUMBER (10)
* PRODUCT_TYPE VARCHAR2 (10 BYTE)
* START_DT DATE
* END_DT DATE
* IS_ACTIVE VARCHAR2 (4 BYTE)
PRODUCT_TYPE_SRCID_PK (PRODUCT_TYPE_SRCID)
PRODUCT_TYPE_SRCID_PK (PRODUCT_TYPE_SRCID)


BL_3NF.CE_SEASONS
P * SEASON_SRCID NUMBER (10)
* SEASON VARCHAR2 (10 BYTE)
SEASON_SRCID_PK (SEASON_SRCID)
SEASON_SRCID_PK (SEASON_SRCID)
BL_3NF.CE_PRODUCTS
P * PRODUCT_SRCID NUMBER (10)
* PRODUCT_NAME VARCHAR2 (10 BYTE)
* PRODUCT_DESCRIPTION VARCHAR2 (40 BYTE)
F * LINE_SRCID NUMBER (10)
F * PRODUCT_TYPE_SRCID NUMBER (10)
* START_DT DATE
* END_DT DATE
* IS_ACTIVE VARCHAR2 (4 BYTE)
PRODUCT_SRCID_PK (PRODUCT_SRCID)
LINE_SRCID_FK (LINE_SRCID)
PRODUCT_TYPE_SRCID_FK (PRODUCT_TYPE_SRCID)
PRODUCT_SRCID_PK (PRODUCT_SRCID)
BL_3NF.CE_PRODUCT_DETAILS
P * PRODUCT_DETAILS_SRCID NUMBER (10)
F * PRODUCT_SRCID NUMBER (10)
HEIGHT VARCHAR2 (10 BYTE)
HIP_GIRTH NUMBER (10)
F BRA_SIZE_SRCID NUMBER (10)
F PANTIES_SIZE_SRCID NUMBER (10)
COLOR VARCHAR2 (40 BYTE)
PRICE NUMBER (10)
PRODUCT_BALANCE NUMBER (10)
* INSERT_DT DATE
UPDATE_DT DATE
PRODUCT_DETAILS_SRCID_PK (PRODUCT_DETAILS_SRCID)
BRA_SIZE_SRCID_FK (BRA_SIZE_SRCID)
PANTIES_SIZE_SRCID_FK (PANTIES_SIZE_SRCID)
PRODUCT_SRCID_FK (PRODUCT_SRCID)
PRODUCT_DETAILS_SRCID_PK (PRODUCT_DETAILS_SRCID)
BL_3NF.CE_PANTIES_SIZE_GRID
P * PANTIES_SIZE_SRCID NUMBER (10)
PANTIES_SIZE_UK VARCHAR2 (10 BYTE)
PANTIES_SIZE_USA VARCHAR2 (10 BYTE)
PANTIES_SIZE_EU VARCHAR2 (10 BYTE)
PANTIES_SIZE_FR VARCHAR2 (10 BYTE)
PANTIES_SIZE_UIE VARCHAR2 (10 BYTE)
PANTIES_SIZE_SRCID_PK (PANTIES_SIZE_SRCID)
PANTIES_SIZE_SRCID_PK (PANTIES_SIZE_SRCID)
BL_3NF.CE_BRA_SIZE_GRID
P * BRA_SIZE_SRCID NUMBER (10)
BRA_SIZE_UK VARCHAR2 (10 BYTE)
BRA_SIZE_USA VARCHAR2 (10 BYTE)
BRA_SIZE_EU VARCHAR2 (10 BYTE)
BRA_SIZE_FR VARCHAR2 (10 BYTE)
BRA_SIZE_UIE VARCHAR2 (10 BYTE)
BRA_SIZE_SRCID_PK (BRA_SIZE_SRCID)
BRA_SIZE_SRCID_PK (BRA_SIZE_SRCID)


BL_3NF.CE_PRODUCT_SALES
P * SALE_SRCID NUMBER (10)
F * RECEIPT_ID NUMBER (10)
F * PRODUCT_DETAILS_SRCID NUMBER (10)
* SALE_SUM NUMBER (10)
* SALE_AMOUNT NUMBER (10)
* INSERT_DT DATE
SALE_SRCID_PK (SALE_SRCID)
PRODUCT_DETAILS_SRCID_FK (PRODUCT_DETAILS_SRCID)
RECEIPT_ID_FK (RECEIPT_ID)
SALE_SRCID_PK (SALE_SRCID)
BL_3NF.CE_CURRENCIES
P * CURRENCY_ID VARCHAR2 (40 BYTE)
* CURRENCY_NAME VARCHAR2 (40 BYTE)
CURRENCY_ID_PK (CURRENCY_ID)
CURRENCY_ID_PK (CURRENCY_ID)
BL_3NF.CE_POSITION_GRADES
P * POSITION_GRADE_SRCID NUMBER (10)
* POSITION_GRADE VARCHAR2 (40 BYTE)
* WORK_PERIOD NUMBER (10)
POSITION_GRADE_SRCID_PK (POSITION_GRADE_SRCID)
POSITION_GRADE_SRCID_PK (POSITION_GRADE_SRCID)
BL_3NF.CE_PAYMENT_METHODS
P * PAYMENT_METHOD_SRCID NUMBER (10)
* PAYMENT_METHOD_NAME VARCHAR2 (40 BYTE)
BANK_NAME VARCHAR2 (40 BYTE)
PAYMENT_METHOD_SRCID_PK (PAYMENT_METHOD_SRCID)
PAYMENT_METHOD_SRCID_PK (PAYMENT_METHOD_SRCID)
BL_3NF.CE_RECEIPTS
P * RECEIPT_ID NUMBER (10)
* RECEIPT_DT DATE
F * STORE_SRCID NUMBER (10)
F * EMPLOYEE_ID NUMBER (10)
F * CUSTOMER_SRCID NUMBER (10)
F * PAYMENT_METHOD_SRCID NUMBER (10)
F * CURRENCY_ID VARCHAR2 (40 BYTE)
* RECEIPT_SUM NUMBER (10)
* INSERT_DT DATE
RECEIPT_ID_PK (RECEIPT_ID)
CURRENCY_ID_FK (CURRENCY_ID)
PAYMENT_METHOD_SRCID_FK (PAYMENT_METHOD_SRCID)
RC_CUSTOMER_SRCID_FK (CUSTOMER_SRCID)
RC_EMPLOYEE_ID_FK (EMPLOYEE_ID)
RC_STORE_SRCID_FK (STORE_SRCID)
RECEIPT_ID_PK (RECEIPT_ID)
BL_3NF.CE_EMPLOYEES
P * EMPLOYEE_ID NUMBER (10)
* FIRST_NAME VARCHAR2 (40 BYTE)
* LAST_NAME VARCHAR2 (40 BYTE)
MANAGER_SRCID NUMBER (10)
F * STORE_SRCID NUMBER (10)
* POSITION_NAME VARCHAR2 (40 BYTE)
F * POSITION_GRADE_SRCID NUMBER (10)
* WORK_PERIOD NUMBER (10)
* EMAIL VARCHAR2 (40 BYTE)
* PHONE VARCHAR2 (40 BYTE)
* START_DT DATE
* END_DT DATE
* IS_ACTIVE VARCHAR2 (4 BYTE)
EMPLOYEE_ID_PK (EMPLOYEE_ID)
POSITION_GRADE_SRCID_FK (POSITION_GRADE_SRCID)
STORE_SRCID_FK (STORE_SRCID)
EMPLOYEE_ID_PK (EMPLOYEE_ID)



Picture 8 3NF model

5 OBJECT PARTITIONING

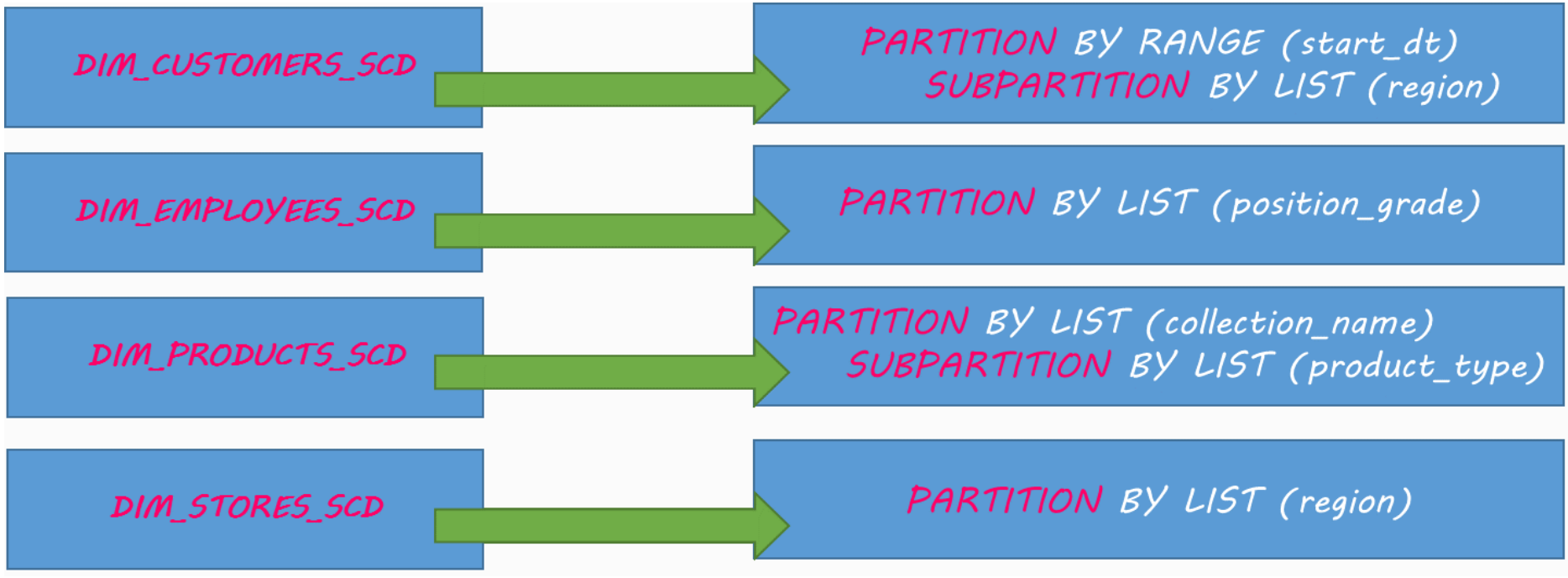
A combination of two methods of data distribution is used to define a table with composite partitioning. Firstly, the table is partitioned by one method of data distribution. Then each section is subdivided into subsections using another method of data distribution. For example, if a range-list is specified for a table, then at the beginning the table is partitioned with keys ranges, and then each section is divided into subsections with the keys lists specified in the second partition. Sections of the table with composite partitioning exist only as metadata and do not provide actual data storage: the subsections of a particular section of a table or an index with composite partitioning are the physical segments of the database where the section data is stored.

In this paper, the partitions of the dimension tables were applied with list and range types of partitions. As indicated in the picture below, the customer dimension was divided into partitions by the year when the client appeared in the database. In turn, each year was divided with sub partitions by the regions. This is very convenient, since it will allow to investigate various cohorts of customers and their activity in time.

Dimension with the employees was divided according to the position grades existing in the company in the timetable. This breakdown will allow to track faster active groups of employees depending on the status.

The product dimension was divided by the collections. Each collection, in turn, is divided by product types, because sales and demand for individual products are usually not tracked, so it's better to take product types into collections as the grain.

The last dimension shops was partitioned by regions. As the sale of a huge holding is usually investigated initially within the region. In a particular region it is easier to find the country of interest than in all dimension.



Picture 9 Partition strategy

6 BUSINESS PROCESSES

In the analysis two business processes will be explored. The first of them is monthly sales of stores within regions, countries and cities, as well as sales by customers, the second - monthly commodity stocks of stores in different geographical sections and per network managers to identify the most valuable personnel.

Three reports are proposed as reports:

1. Sales analysis.

This table will be designed to provide general information about the functioning stores, sales, the number of products in stock, and the number of receipts. The lowest level of granularity will be represented by regions, and then DRILL-DOWN technology will make it possible to deepen the detailed analysis of analytics to countries and cities. The highest level of hierarchy will be presented by shops with a specific address in the city.

The following parameters will be presented in the table: the turnover for the previous period - the number of months of the previous period can be determined by specifying the corresponding value in the "Comparison period" field, the current period can be determined in the "Analysis period" field. The completion percentage will show what percentage of the revenue of the previous period is reached for the current period. The indicator quantity of goods determines the commodity remains, and the number of receipts shows how many sales were made for the analyzed period.

2. Analysis of receipts.

This report will allow user to view value of the main indicators per the metrics selected in the ListBox. In each ListBox, user can select multiple values at the same time. The analysis period can be determined by analogy with the previous report. The choice of month and year is provided using the ListBox.

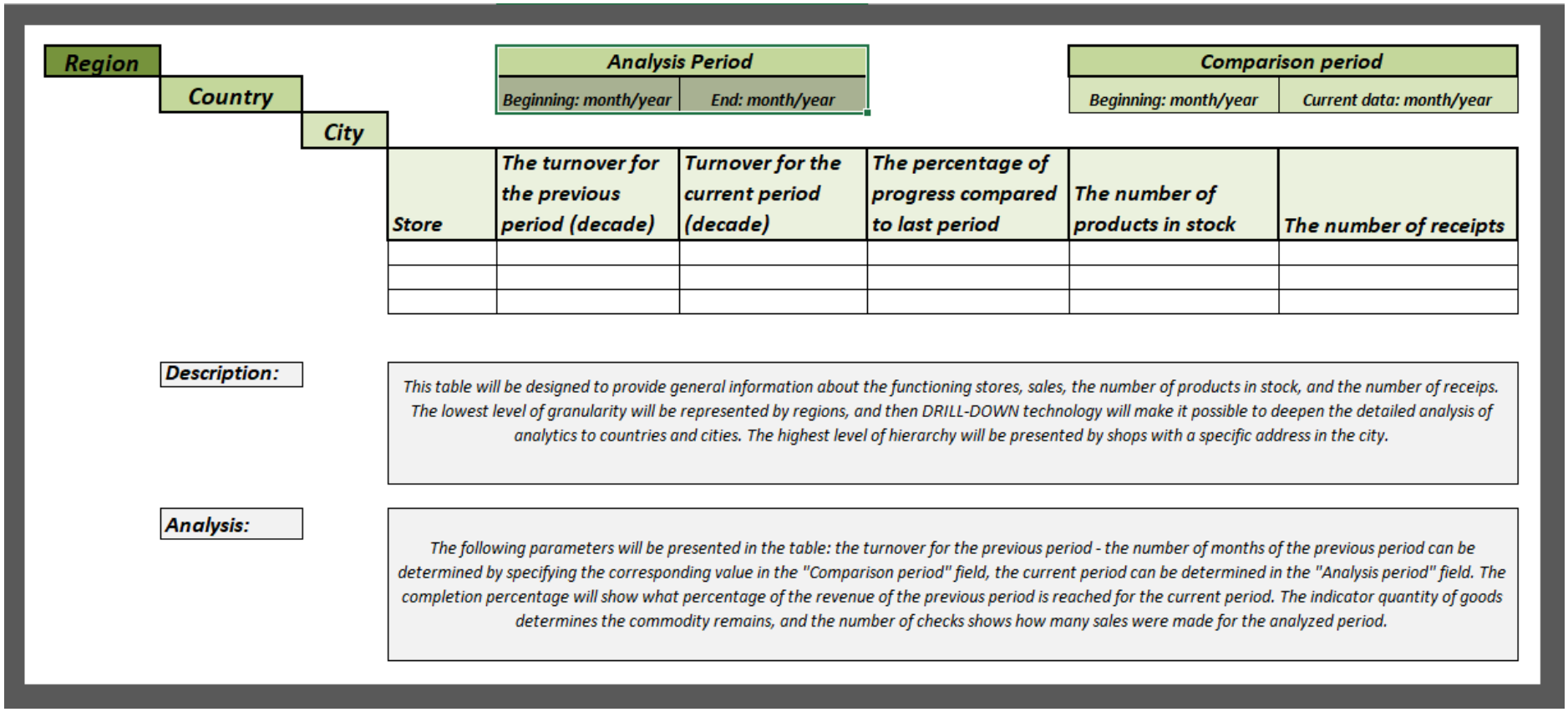
3. Load analysis.

This analysis is used to investigate customers' influx during working days and working hours of the day what will be chosen for analysis. This analysis allows user to see the most favorable working periods for the promotions, as well as the time when it is necessary to have more staff in stores.

4. Data generation.

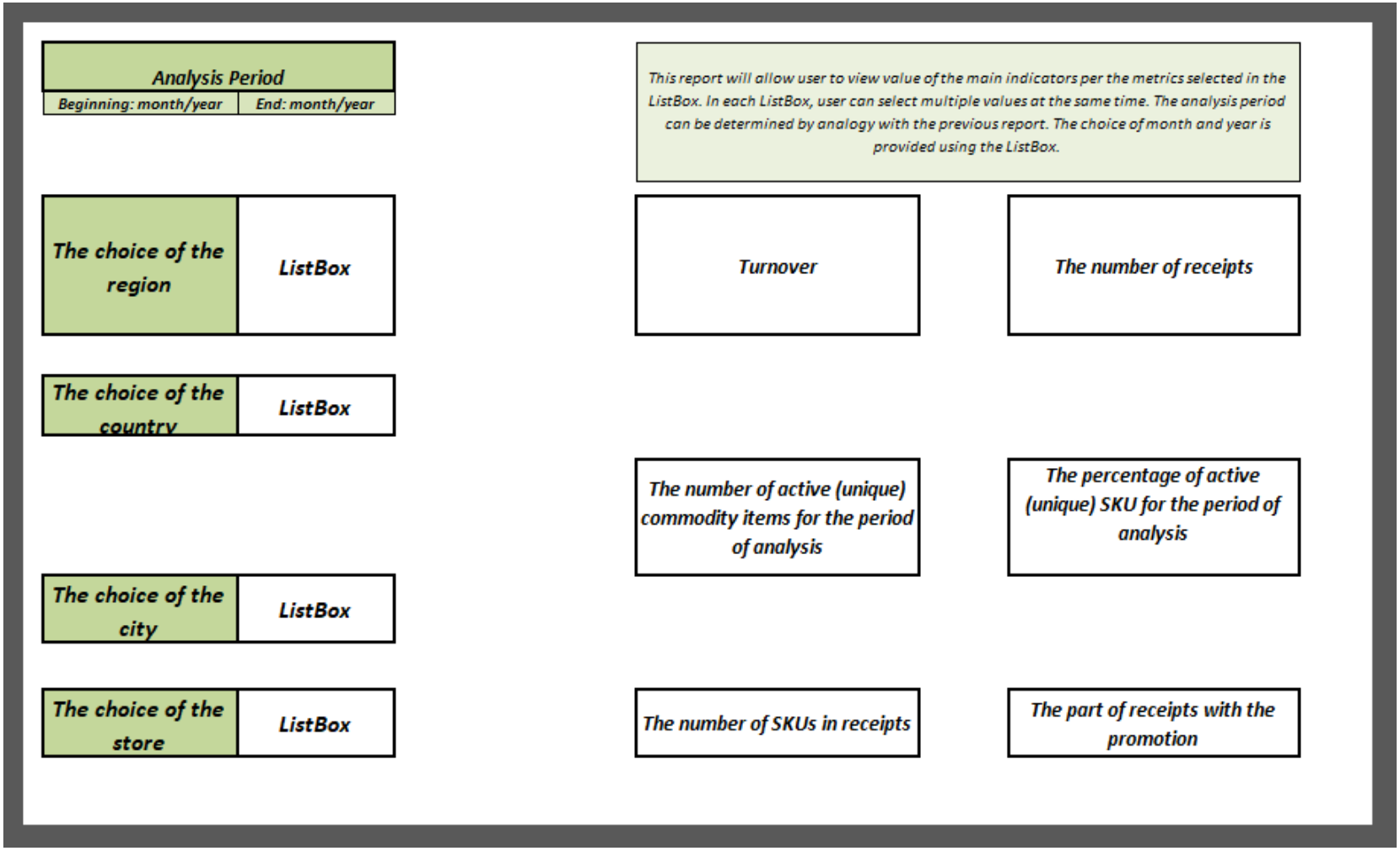
This tab contains tables with generated data to plot graphs in the "Load Analysis" report.

6.1 SALES ANALYSIS



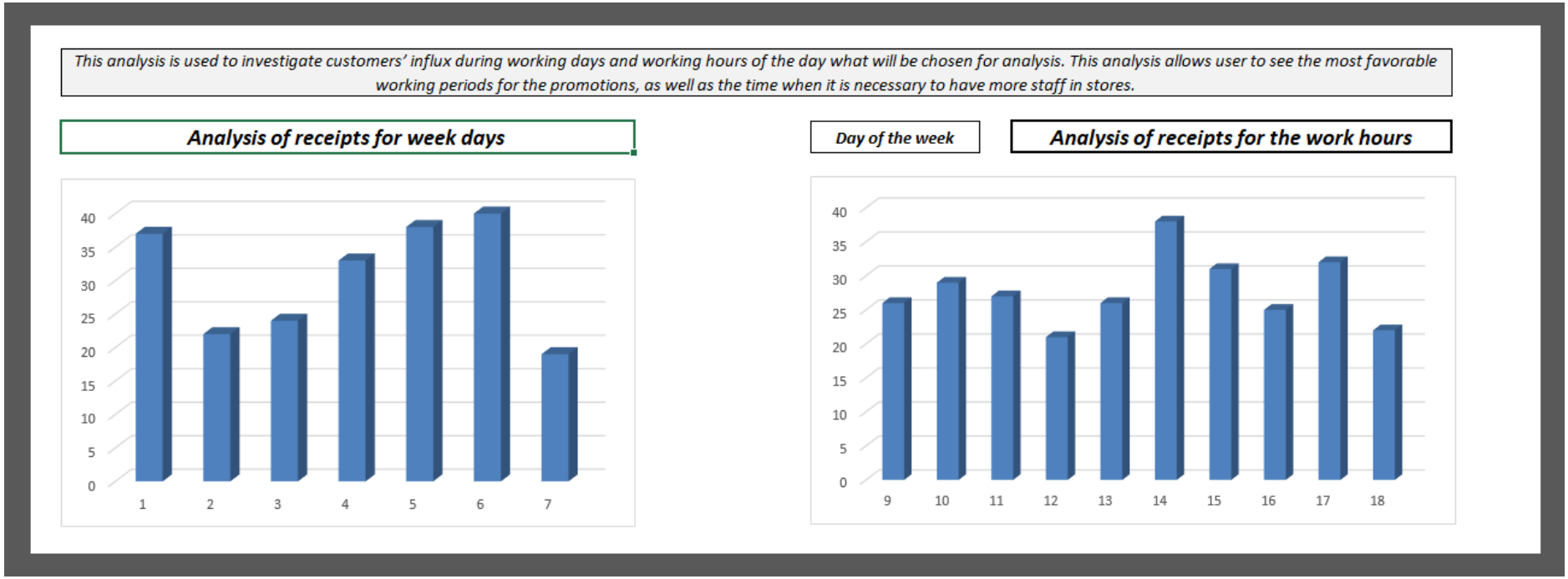
Picture 10 Sales analysis

6.2 ANALYSIS OF RECEIPTS



Picture 11 Analysis of receipts

6.3 LOAD ANALYSIS



Picture 12 Load analysis

7 DATA MODELLING

7.1 DETAILING DIAGRAMS FOR 3NF AND STAR/SNOWFLAKE LAYERS

There were created seven dimensions. They are described in “Business Description #4” document. Several dimensions have a SCD2 type. It was chosen due to It is the next dimensions:

1. dim_customers_scd;
2. dim_employees_scd;
3. dim_payment_methods_scd;
4. dim_products_scd;
5. dim_promotions_scd;

Dim_time_day has calendar type. Dim_stores has a SCD1 type, because it is not necessary to have history about closed stores.

7.2 TEXTUAL DESCRIPTION OF LAYERS OF DATA WAREHOUSE

In this work was used a two-layer architecture.

7.2.1 Source layer

There were used flat files and html-code as data sources.

7.2.2 Data staging layer

The data stored to sources should be extracted, cleansed to remove inconsistencies and fill gaps, and integrated to merge heterogeneous sources into one common schema.

In this work will be used the following data staging layers step by step.

7.2.2.1 Cleansing layer

This layer was used for data cleansing, filtering wrong data, replace missing values with singletons and performing transformations like code lookups or currency conversions. As the Staging Area, the Cleansing Area contains only data of the last delivery, and data from different sources is not integrated.

7.2.2.2 3NF layer

Here reducing data redundancy and improving data integrity were made. Also, the process of simplifying the design of a database, so that it achieves the optimal structure composed of atomic elements, was achieved on this layer.

7.2.2.3 Dimensional model layer

The Star schema was chosen as a dimensional model for business processes description.

The main reasons:

1. Simple structure;
2. Absence of a big number of tables to join;
3. Denormalized tables are not too large in a specific case of this task;
4. Widely support by a large number of business intelligence tools.

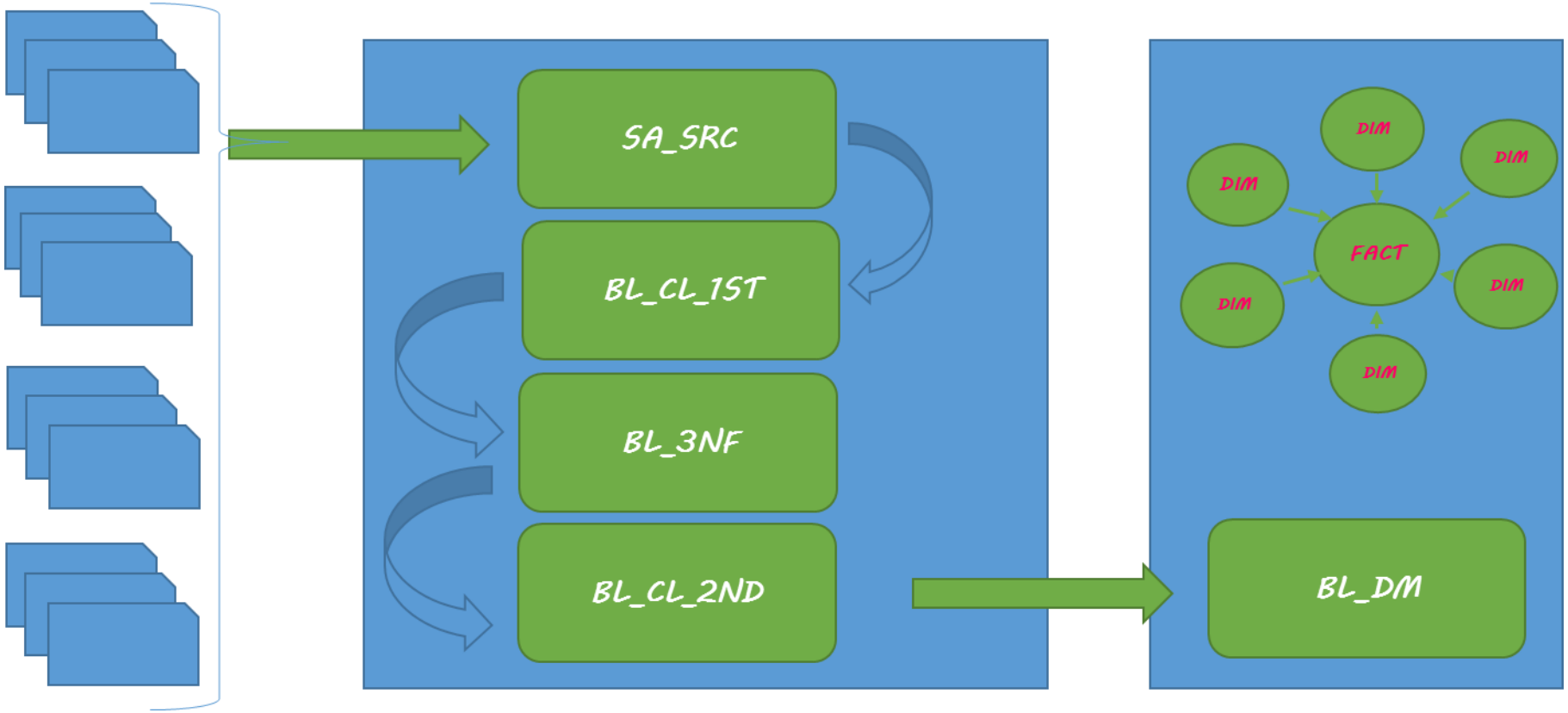
7.2.3 Data warehouse layer

Information is stored to one logically centralized single repository: a data warehouse. The data warehouse can be directly accessed, but it can also be used as a source for creating data marts, which partially replicate data warehouse contents and are designed for specific departments. This layer is added after finishing previous steps.

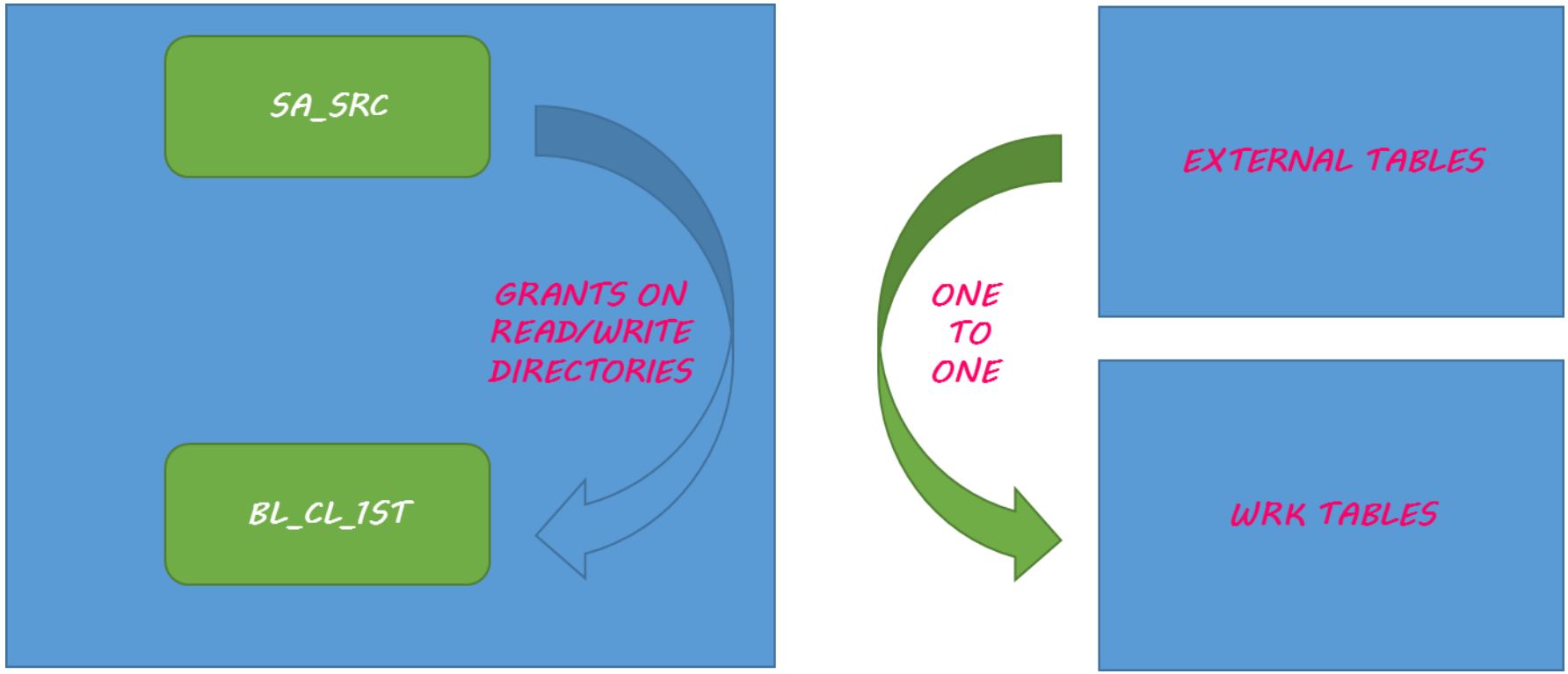
7.2.4 Analysis layer

In this layer, integrated data is efficiently and flexibly accessed to issue reports, dynamically analyze information, and simulate hypothetical business scenarios. Template of this layer is represented in “Retail Analysis” document.

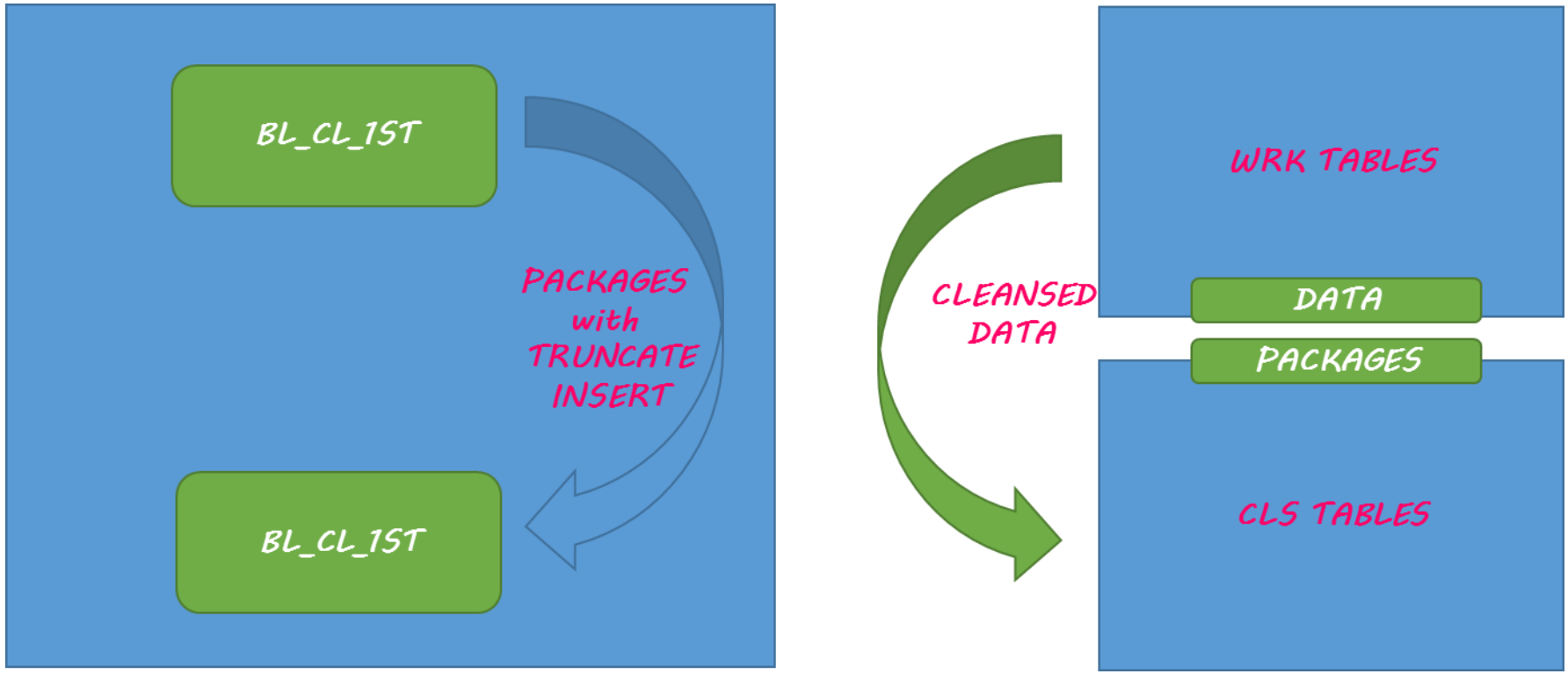
7.3 VISUAL DESCRIPTION OF LAYERS OF DATA WAREHOUSE



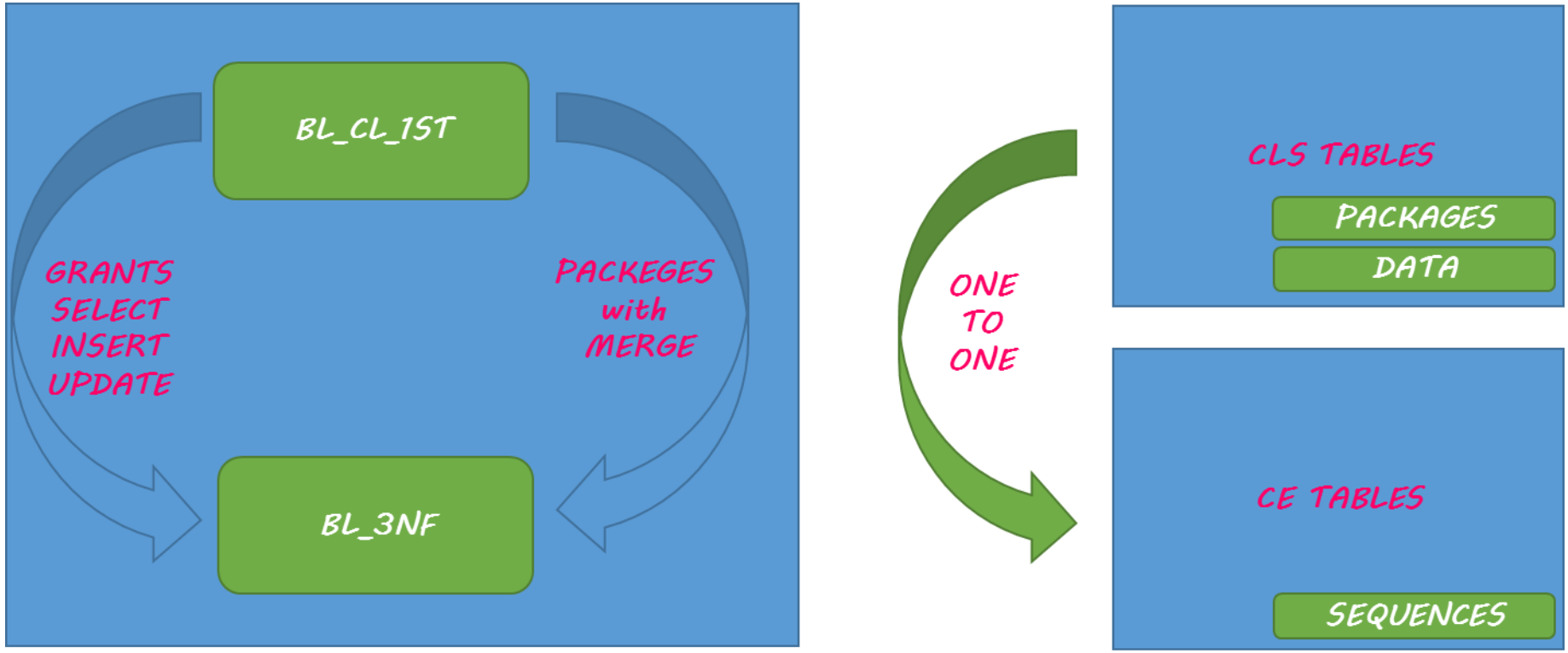
Picture 13 Layers model



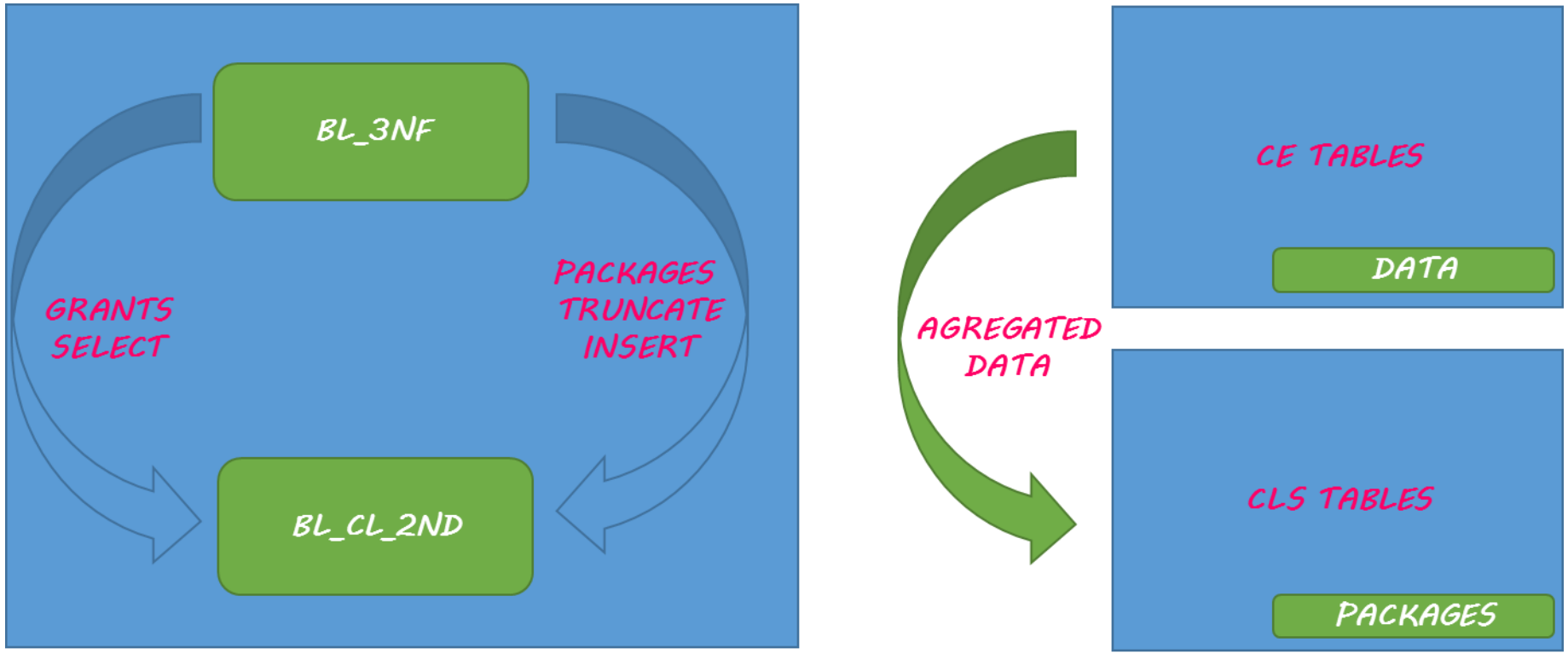
Picture 14 SA SRC and BL CL 1ST



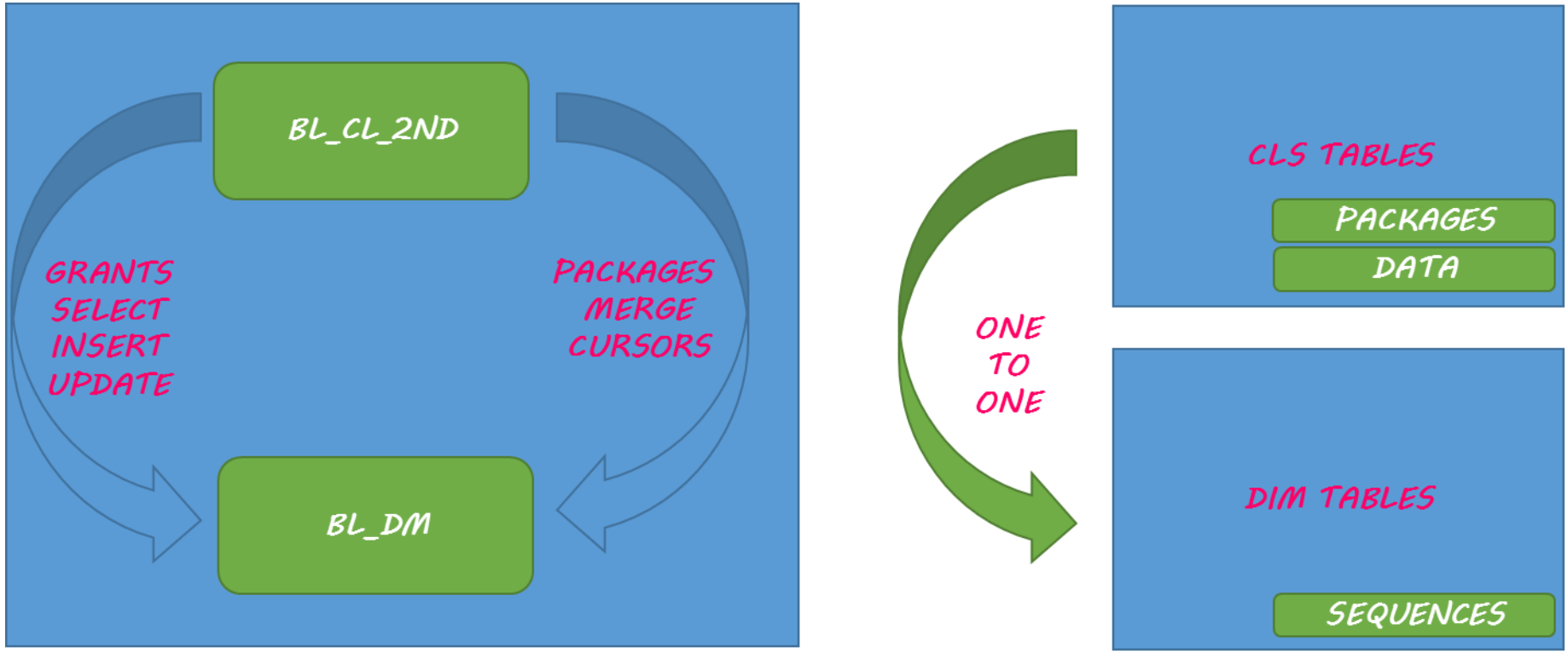
Picture 15 BL CL 1ST



Picture 16 BL CL 1ST and BL 3NF



Picture 17 BL 3NF and BL CL 2nd

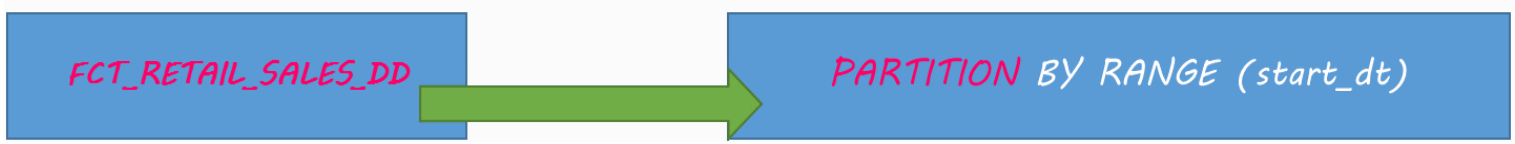


Picture 18 BL CL 2nd and BL DM

8 FACT TABLE PARTITIONING STRATEGY

Among the available partitioning methods, the above-described RANGE was applied in to fact table.

This partitioning method RANGE-is most convenient for the fact table. It will be divided into year's partitions, and every year, in turn, will contain partitions with specific regions where sales are conducted. As the sales network is quite extensive, the granularity in the partitions to the level of the cities is not expedient. For business, it will be much more convenient to track sales within a specific region.



Picture 19 Partition strategy

9 STRATEGY OF PARALLEL LOAD

Parallel data loading can improve the performance of DML operators performed radically. With consecutive execution of operations, each subsequent must wait for the end of the preceding one. Parallel execution helps to avoid downtime and allows the next in turn operation to begin processing data that has already been processed by the first operation, even though the first operation was not completed completely.

When loading the generated data, this possibility can be applied to uploading data to the BL_DM layer. Since the SELECT and INSERT operations for each table are used for this table, PARALLEL LOAD will allow to select data for insertion and load the already selected data simultaneously, which in theory should improve the performance of our queries.

Let's check the time plan for the query execution before applying the hint on the fact table.

```
Plan hash value: 4253285374

-----------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                    | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |                         |     1 |   185 | 13269   (1)| 00:00:01 |       |       |
|   1 |  LOAD TABLE CONVENTIONAL | FCT_RETAIL_SALES_DD     |       |       |            |          |       |       |
|   2 |   SEQUENCE               | FCT_RETAIL_SALES_DD_SEQ |       |       |            |          |       |       |
|*  3 |    HASH JOIN             |                         |     1 |   185 | 13269   (1)| 00:00:01 |       |       |
|   4 |     NESTED LOOPS         |                         |     1 |   159 | 12177   (1)| 00:00:01 |       |       |
|*  5 |      HASH JOIN           |                         |     1 |   146 | 12177   (1)| 00:00:01 |       |       |
|*  6 |       HASH JOIN          |                         |     1 |   120 | 10539   (1)| 00:00:01 |       |       |
|*  7 |        HASH JOIN         |                         |     1 |    94 |  8901   (1)| 00:00:01 |       |       |
|   8 |         PARTITION RANGE ALL|                       |     1 |    26 |  3375   (1)| 00:00:01 |     1 |    11 |
|   9 |          PARTITION LIST ALL|                       |     1 |    26 |  3375   (1)| 00:00:01 |     1 |     6 |
|  10 |           TABLE ACCESS FULL| DIM_CUSTOMERS_SCD     |     1 |    26 |  3375   (1)| 00:00:01 |     1 |    66 |
|  11 |        TABLE ACCESS FULL | CLS_FCT_RETAIL_SALES_DD | 2001K | 129M  |  5520   (1)| 00:00:01 |       |       |
|  12 |       PARTITION LIST ALL |                         |     1 |    26 |  1638   (1)| 00:00:01 |     1 |     6 |
```

```
2,001,998 rows inserted.

Elapsed: 00:06:53.634
```

Picture 20 Explain plan

The query execution plan using - + PARALLEL (4).

```
Plan hash value: 2766409812

-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                    | Rows  | Bytes | Cost (%CPU)| Time
-----------------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                   |                         |     1 |   185 |  3683   (1)| 00:00:01
|   1 |  LOAD TABLE CONVENTIONAL           | FCT_RETAIL_SALES_DD     |       |       |            |
|   2 |   SEQUENCE                         | FCT_RETAIL_SALES_DD_SEQ |       |       |            |
|   3 |    PX COORDINATOR                  |                         |       |       |            |
|   4 |     PX SEND QC (RANDOM)            | :TQ10006                |     1 |   185 |  3683   (1)| 00:00:01
|*  5 |      HASH JOIN                     |                         |     1 |   185 |  3683   (1)| 00:00:01
|   6 |       PX RECEIVE                   |                         |     1 |   159 |  3380   (1)| 00:00:01
|   7 |        PX SEND BROADCAST           | :TQ10005                |     1 |   159 |  3380   (1)| 00:00:01
|   8 |         BUFFER SORT                |                         |   902 |  126K|            |
|   9 |          NESTED LOOPS              |                         |     1 |   159 |  3380   (1)| 00:00:01
|* 10 |           HASH JOIN                |                         |     1 |   146 |  3380   (1)| 00:00:01
|  11 |            PX RECEIVE              |                         |     1 |   120 |  2925   (1)| 00:00:01
|  12 |             PX SEND HYBRID HASH    | :TQ10003                |     1 |   120 |  2925   (1)| 00:00:01
```

```
2,001,998 rows inserted.

Elapsed: 00:09:02.015
```

Picture 21 Explain plan

You can see that despite the decrease in the COST parameter, the parallel in this case had the opposite effect - the query execution time increased by two minutes, which is very significant. In this case, the use of a parallel is impractical. Let's try to increase the number of parallel processes to 20.

The query execution plan using - + PARALLEL (10).

```
Plan hash value: 2766409812

-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                    | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                   |                         |     1 |   185 |  1473   (1)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL           | FCT_RETAIL_SALES_DD     |       |       |            |          |
|   2 |   SEQUENCE                         | FCT_RETAIL_SALES_DD_SEQ |       |       |            |          |
|   3 |    PX COORDINATOR                  |                         |       |       |            |          |
|   4 |     PX SEND QC (RANDOM)            | :TQ10006                |     1 |   185 |  1473   (1)| 00:00:01 |
|*  5 |      HASH JOIN                     |                         |     1 |   185 |  1473   (1)| 00:00:01 |
|   6 |       PX RECEIVE                   |                         |     1 |   159 |  1352   (1)| 00:00:01 |
|   7 |        PX SEND BROADCAST           | :TQ10005                |     1 |   159 |  1352   (1)| 00:00:01 |
```

```
2,001,998 rows inserted.

Elapsed: 00:20:18.325
```

Picture 22 Explain plan

This request was carried out for 20 minutes.

Obviously, the time for downloading is only increased using PARALLEL on the fact table. We test the hint on the DIMENSION table DIM_PRODUCTS_SCD, because it is the largest of the DIMENSION tables.

The query execution plan without application is + PARALLEL.

```
Plan hash value: 24526241

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------
|   0 | MERGE STATEMENT              |                   |  6000 |    14M|       |   182   (2)| 00:00:01 |
|   1 |  MERGE                       | DIM_PRODUCTS_SCD  |       |       |       |            |          |
|   2 |   VIEW                       |                   |       |       |       |            |          |
|   3 |    SEQUENCE                  | DIM_PRODUCTS_SEQ  |       |       |       |            |          |
|*  4 |     HASH JOIN RIGHT OUTER    |                   |  6000 |    14M|       |   182   (2)| 00:00:01 |
|   5 |      PARTITION LIST ALL      |                   |     1 |  1251 |       |     2   (0)| 00:00:01 |
|   6 |       PARTITION LIST ALL     |                   |     1 |  1251 |       |     2   (0)| 00:00:01 |
|   7 |        TABLE ACCESS FULL     | DIM_PRODUCTS_SCD  |     1 |  1251 |       |     2   (0)| 00:00:01 |
|   8 |      VIEW                    |                   |  6000 |  7183K|       |   180   (2)| 00:00:01 |
|   9 |       MINUS                  |                   |       |       |       |            |          |
|  10 |        SORT UNIQUE           |                   |  6000 |   621K|   920K|            |          |

6,000 rows merged.

Elapsed: 00:00:01.440
```

Picture 23 Explain plan

The query execution plan using - + PARALLEL (4).

```
Plan hash value: 3260348670

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                        | Name             | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | MERGE STATEMENT                  |                  |  6000 |    14M|       |    15  (20)| 00:00:01 |       |       |
|   1 |  MERGE                           | DIM_PRODUCTS_SCD |       |       |       |            |          |       |       |
|   2 |   VIEW                           |                  |       |       |       |            |          |       |       |
|   3 |    SEQUENCE                      | DIM_PRODUCTS_SEQ |       |       |       |            |          |       |       |
|   4 |     PX COORDINATOR               |                  |       |       |       |            |          |       |       |
|   5 |      PX SEND QC (RANDOM)         | :TQ10003         |  6000 |    14M|       |    15  (20)| 00:00:01 |       |       |
|*  6 |       HASH JOIN RIGHT OUTER BUFFERED|               |  6000 |    14M|       |    15  (20)| 00:00:01 |       |       |
|   7 |        PX PARTITION LIST ALL     |                  |     1 |  1251 |       |     2   (0)| 00:00:01 |     1 |     3 |
|   8 |         PX PARTITION LIST ALL    |                  |     1 |  1251 |       |     2   (0)| 00:00:01 |     1 |     2 |
|   9 |          TABLE ACCESS FULL       | DIM_PRODUCTS_SCD |     1 |  1251 |       |     2   (0)| 00:00:01 |     1 |     6 |
|  10 |        PX RECEIVE                |                  |  6000 |  7183K|       |    13  (24)| 00:00:01 |       |       |
```

```
6,000 rows merged.

Elapsed: 00:00:01.204
```

Picture 24 Explain plan

Dimension tables are much smaller than the fact table and are loaded quite quickly, so the use of parallel loading on them is also inappropriate.
Acceleration of the request for a millisecond will not make a significant difference to the performance in the projected DWH.
Thus, the parallel boot strategy will not be applied in this work.

10 REPORT LAYOUTS

10.1 3NF-LAYER

Execute the query with the application of analytical functions.

Let's analyze sales for 2018 with grouping by quarter, month and day.

```
SELECT DECODE(GROUPING_ID(to_char(receipt_dt,'YYYY'),
                          upper(TO_CHAR(receipt_dt,'YYYY')) || '-' || 'Q' || TO_CHAR(receipt_dt,'Q'),
                          upper(TO_CHAR(receipt_dt,'YYYY') || '-' || TO_CHAR(receipt_dt,'Mon') ), receipt_dt), 7,
                          'GRAND TOTAL FOR ' || to_char(receipt_dt,'YYYY'), ' ') AS year,
       DECODE(GROUPING_ID(to_char(receipt_dt,'YYYY'),
                          upper(TO_CHAR(receipt_dt,'YYYY')) || '-' || 'Q' || TO_CHAR(receipt_dt,'Q'),
                          upper(TO_CHAR(receipt_dt,'YYYY') || '-' || TO_CHAR(receipt_dt,'Mon') ), receipt_dt), 3,
                          'GRAND TOTAL FOR ' || upper(TO_CHAR(receipt_dt,'YYYY')) || '-' || 'Q' ||
TO_CHAR(receipt_dt,'Q'), ' ') AS quarter,
       DECODE(GROUPING_ID(to_char(receipt_dt,'YYYY'),
                          upper(TO_CHAR(receipt_dt,'YYYY')) || '-' || 'Q' || TO_CHAR(receipt_dt,'Q'),
                          upper(TO_CHAR(receipt_dt,'YYYY') || '-' || TO_CHAR(receipt_dt,'Mon') ), receipt_dt), 1,
                          'GRAND TOTAL FOR ' || upper(TO_CHAR(receipt_dt,'YYYY') || '-' || TO_CHAR(receipt_dt,'Mon')
), ' ') AS month,
       DECODE(GROUPING(receipt_dt), 1, ' ', receipt_dt) AS day,
       TO_CHAR(SUM(receipt_sum_usd), '9,999,999,999') as sales
FROM   ce_receipts dt
```

```
WHERE  to_char(receipt_dt,'YYYY') = 2018
GROUP BY ROLLUP(
                to_char(receipt_dt,'YYYY'),
                upper(TO_CHAR(receipt_dt,'YYYY')) || '-' || 'Q' || TO_CHAR(receipt_dt,'Q'),
                upper(TO_CHAR(receipt_dt,'YYYY') || '-' || TO_CHAR(receipt_dt,'Mon') ),
                receipt_dt
);
```

	YEAR	QUARTER	MONTH	DAY	SALES
363				15-OCT-18	98,632,990
364				16-OCT-18	101,978,967
365				17-OCT-18	99,178,589
366				18-OCT-18	99,693,233
367				19-OCT-18	101,251,898
368				20-OCT-18	99,724,227
369				21-OCT-18	97,660,012
370				22-OCT-18	95,913,192
371				23-OCT-18	100,312,667
372				24-OCT-18	96,909,652
373				25-OCT-18	99,616,695
374				26-OCT-18	101,978,782
375				27-OCT-18	104,302,014
376				28-OCT-18	101,581,387
377				29-OCT-18	103,801,735
378				30-OCT-18	102,354,341
379				31-OCT-18	99,523,147
380			GRAND TOTAL FOR 2018-OCT		3,104,654,916
381		GRAND TOTAL FOR 2018-Q			9,238,938,174
382	GRAND TOTAL FOR 2018				##############
383					##############

All Rows Fetched: 383 in 2.394 seconds

```
PLAN_TABLE_OUTPUT
1  Plan hash value: 3141232678
2
3  ------------------------------------------------------------------------------------
4  | Id  | Operation            | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
5  ------------------------------------------------------------------------------------
6  |   0 | SELECT STATEMENT     |             |  1001 | 13013 |  4716   (1)| 00:00:01 |
7  |   1 |  SORT GROUP BY ROLLUP|             |  1001 | 13013 |  4716   (1)| 00:00:01 |
8  |*  2 |   TABLE ACCESS FULL  | CE_RECEIPTS | 20020 |   254K|  4715   (1)| 00:00:01 |
9  ------------------------------------------------------------------------------------
10
11 Predicate Information (identified by operation id):
12 ---------------------------------------------------
13
14    2 - filter(TO_NUMBER(TO_CHAR(INTERNAL_FUNCTION("RECEIPT_DT"),'YYYY'))=201
15              8)
```

Picture 25 Explain plan

10.2 DM-LAYER

Perform a similar query on the DM layer.

```
SELECT DECODE(GROUPING_ID(dt.year, dt.quarter_year, dt.month_year, event_dt), 7, 'GRAND TOTAL FOR ' || dt.year, ' ')
AS year,
       DECODE(GROUPING_ID(dt.year, dt.quarter_year, dt.month_year, event_dt), 3, 'GRAND TOTAL FOR ' ||
dt.quarter_year, ' ') AS quarter,
       DECODE(GROUPING_ID(dt.year, dt.quarter_year, dt.month_year, event_dt), 1, 'GRAND TOTAL FOR ' ||
dt.month_year, ' ') AS month,
       DECODE(GROUPING(event_dt), 1, ' ', event_dt) AS day,
       TO_CHAR(SUM(fct.tot_sale_sum), '9,999,999,999') as sales
FROM   fct_retail_sales_dd fct,
       dim_time_day dt
WHERE  dt.date_dt = fct.event_dt
  AND  dt.year = 2018
GROUP BY ROLLUP(
                dt.year,
                dt.quarter_year,
                dt.month_year,
                event_dt
);
```

BL_DM_sales.sql

SQL Worksheet | History

Worksheet | Query Builder

```
        TO_CHAR(SUM(fct.tot_sale_sum), '9,999,999,999') as sales
```

Query Result

SQL | All Rows Fetched: 383 in 0.675 seconds

	YEAR	QUARTER	MONTH	DAY	SALES
363				15-OCT-18	98,632,990
364				16-OCT-18	101,978,967
365				17-OCT-18	99,178,589
366				18-OCT-18	99,693,233
367				19-OCT-18	101,251,898
368				20-OCT-18	99,724,227
369				21-OCT-18	97,660,012
370				22-OCT-18	95,913,192
371				23-OCT-18	100,312,667
372				24-OCT-18	96,909,652
373				25-OCT-18	99,616,695
374				26-OCT-18	101,978,782
375				27-OCT-18	104,302,014
376				28-OCT-18	101,581,387
377				29-OCT-18	103,801,735
378				30-OCT-18	102,354,341
379				31-OCT-18	99,523,147
380			GRAND TOTAL FOR 2018-OCT		3,104,654,916
381		GRAND TOTAL FOR 2018-Q4			9,238,938,174
382	GRAND TOTAL FOR 2018				###############
383					###############

All Rows Fetched: 383 in 0.675 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 3413384702

-----------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                     |   589K|    24M|       | 12849   (1)| 00:00:01 |       |       |
|   1 |  SORT GROUP BY ROLLUP         |                     |   589K|    24M|    29M| 12849   (1)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                   |                     |   589K|    24M|       |  6378   (1)| 00:00:01 |       |       |
|   3 |    PART JOIN FILTER CREATE    | :BF0000             |   589K|    24M|       |  6378   (1)| 00:00:01 |       |       |
|*  4 |     TABLE ACCESS FULL         | DIM_TIME_DAY        |   365 | 10950 |       |    68   (0)| 00:00:01 |       |       |
|   5 |    PARTITION RANGE JOIN-FILTER|                     |  2001K|    24M|       |  6305   (1)| 00:00:01 |:BF0000|:BF0000|
|   6 |     TABLE ACCESS FULL         | FCT_RETAIL_SALES_DD |  2001K|    24M|       |  6305   (1)| 00:00:01 |:BF0000|:BF0000|
-----------------------------------------------------------------------------------------------------------------------------
```

Picture 26 Explain plan

Conclusion.

Despite the fact that the COST parameter in 3NF is much lower, the query in 3NF was executed in 2.39 seconds, and in DWH - for 0.68, what demonstrates the efficiency of DWH.