|  |
| --- |
| Business Template  **Online sTOREs** |
|  |

Contents

[1.1 Business background 3](#_Toc96029391)

[1.2 Problems. Current Situation 3](#_Toc96029392)

[1.3 Benefits from implementing a Data Warehouse 3](#_Toc96029393)

[2.1 Dataset description 3](#_Toc96029394)

[2.2 Four-step dimensional design process 4](#_Toc96029395)

[3.0 Logical scheme 6](#_Toc96029391)

[4.0 Data flow 6](#_Toc96029391)

[5.0 Partitioning strategy](#_Toc96029391) 9

# 

# Business Description

## Business background

There are 2 online shops: ExpressBazaar and GetQuickStore.

ExpressBazaar is an online retail shop based in China that offers products to international online buyers. GetQuickStore is also online retail shop based in Poland.

CEO of ExpressBazaar want to increase revenue so this companies were merged. All transactions of ExpressBazaar were saved in RetailCRM and all sales of GetQuickStore were written in AmoCRM.

## Problems. Current Situation

In our situation there is a problem that top managers don’t have any analytics about sales revenue, client’s profile, sales trends.

Managers of the company see the urgent necessity of creating a company database and analytics, which will collect and analyze all the information about transactions that are made within online shops.

## Benefits from implementing a Data Warehouse

Using of data warehouse can help you with the problems described above. Implementing a data warehouse can answer you the following questions:

* Products and their sales in dynamics. How much is the revenue for each month and for each year? Is there any differences between sales in different months?
* We can understhand if we need to start marketing company, promotion to increase attractiveness and sales.
* Client’s profile: which payment method is preferred for every gender?
* What is the most popular product?
* How much is the revenue for each month and for each year?
* And many other.

# DIMENSIONS OF A BUSINESS

## DATASET DESCRIPTION

The dataset was found here https://www.kaggle.com/vipin20/transaction-data/version/1.

The Trasaction\_data.csv file contains data with about 1 mln of the real supermarket transactions. This dataset was divided into 2 datasets. 1 dataset contains information about 1 shop and 1 more dataset contains transactions of the 2nd shop.

Each row in dataset represents one transaction per 1 item (payment). The columns describe different attributes about transactions including date and time of transaction, Product ID, Product Name, Quantity, Price Per Item, Country where transaction was done and other. Data such as payment method, sales channels, employee’s and customer’s info were also added.

This data model includes 1 fact table and 7 dimension tables.

## FOUR-STEP DIMENSIONAL DESIGN PROCESS

STEP 1: SELECT THE BUSINESS PROCESS

A business process in our situation is receiving payments for sales.

STEP 2: DECLARE THE GRAIN

Fact tables are designed to a low level of uniform detail (referred to as "granularity" or "grain"), meaning facts can record events at a very atomic level. The grain in our model is one row per product sold on a sales transaction (payment).

STEP 3: IDENTIFY THE DIMENSIONS

The dimensions typically can easily be identified as they represent the “who, what, where, when, why, and how” associated with the event.

Questions and answers will be next:

|  |  |  |
| --- | --- | --- |
| Questions | Dimensions | Answers |
| Who? | Customer\_geo  Employees | Customers dimension table describes location data, such as Customers’ country, city, address.  Employee dimension table describes employees in our stores. |
| What? | Products | It’s used to store details about products we’ll offer to our clients. |
| Where? | Stores  Channels | Stores dimension table describes stores where customers can buy products (ExpressBazaar or GetQuickStore).  Channels dimension table describes Channels where products can be bought (Ebay, Amazon). |
| When? | Date | Time dimension table describes time at the lowest level of time granularity for which payments can be made. |
| How? | Payment\_type | How will customer pay? (By card or using PayPal or Qiwi). |

STEP 4: IDENTIFY THE FACTS

Examples of fact data include measurements as sales price and sales quantity.

Now we get Star layer. Our star schema is denormalized.

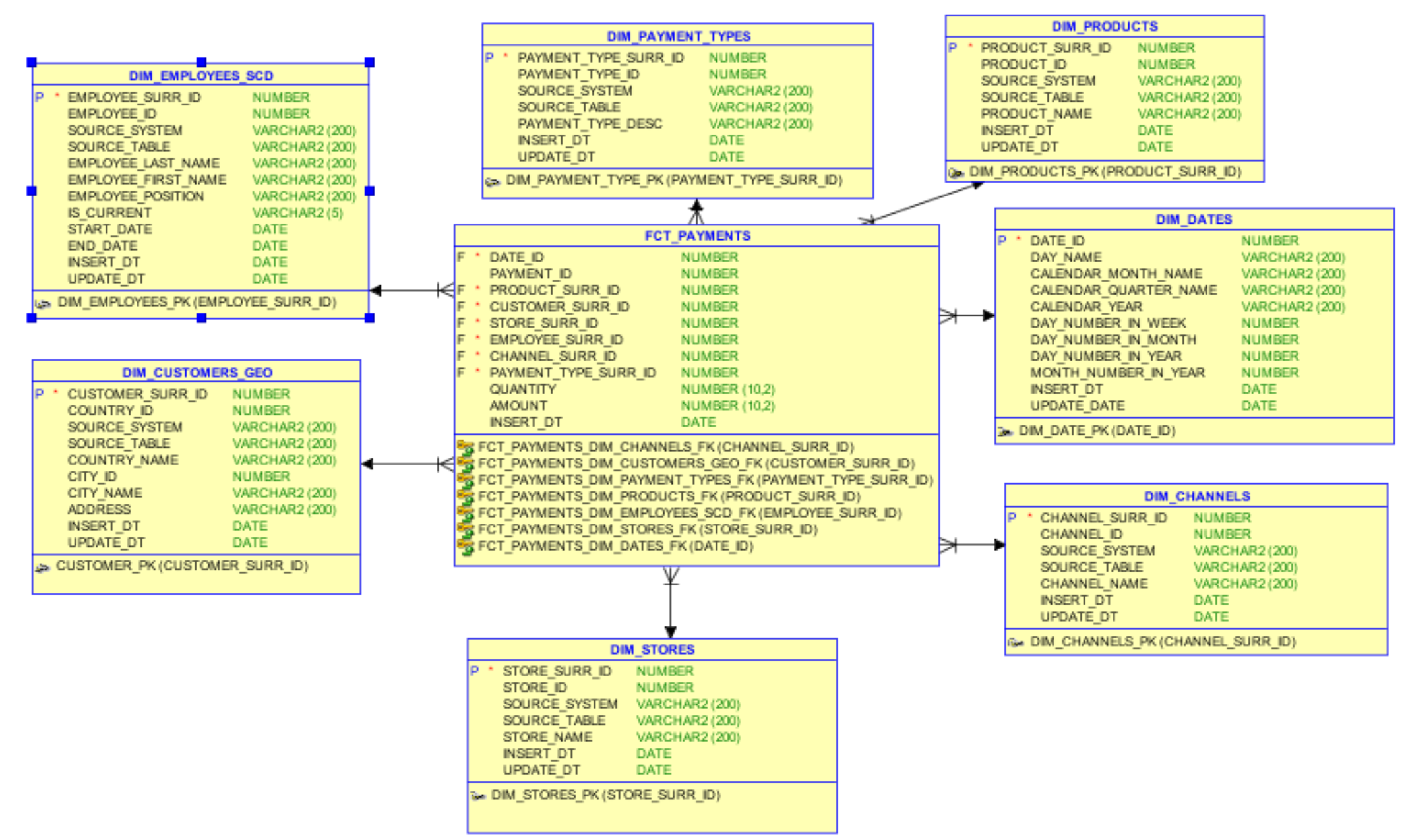
Star schemas divide data into facts and dimensions. Facts are the measurements of some event such as a sale and are typically numbers. Dimensions are the categories you use to identify facts, such as date, location, and product.

We have 2 fields in each table \_id and \_surr\_id, where \_id is the surrogate Key generated in the previous system and \_surr\_id is the surrogate Key generated at the stage of loading into the Star schema. This key is a simple primary key.

Employees Dimensions Table has SCD2 Type. A Type 2 SCD retains the full history of values. When the value of a chosen attribute (employee position) changes, the current record is closed. A new record is created with the changed data values and this new record becomes the current record.

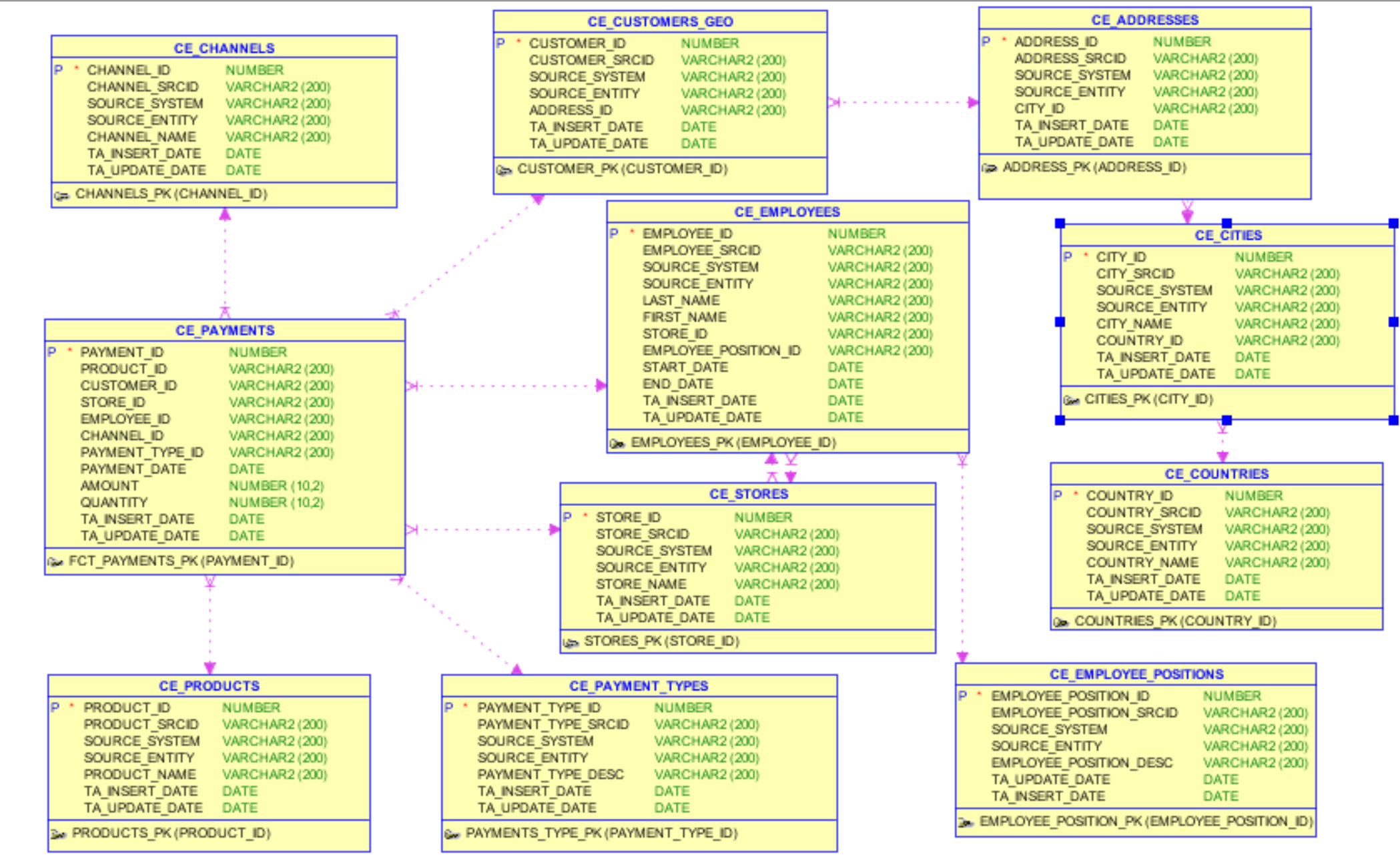
The date dimension has hierarchies that goes from day to week, month, quarter and year.

Star schema:



Third Normal Form design seeks to minimize data redundancy and avoid anomalies in data insertion, updates and deletion. This schema is normalized.

3NF Schema:



# Logical SCHEMA

Logical model of DWH load:

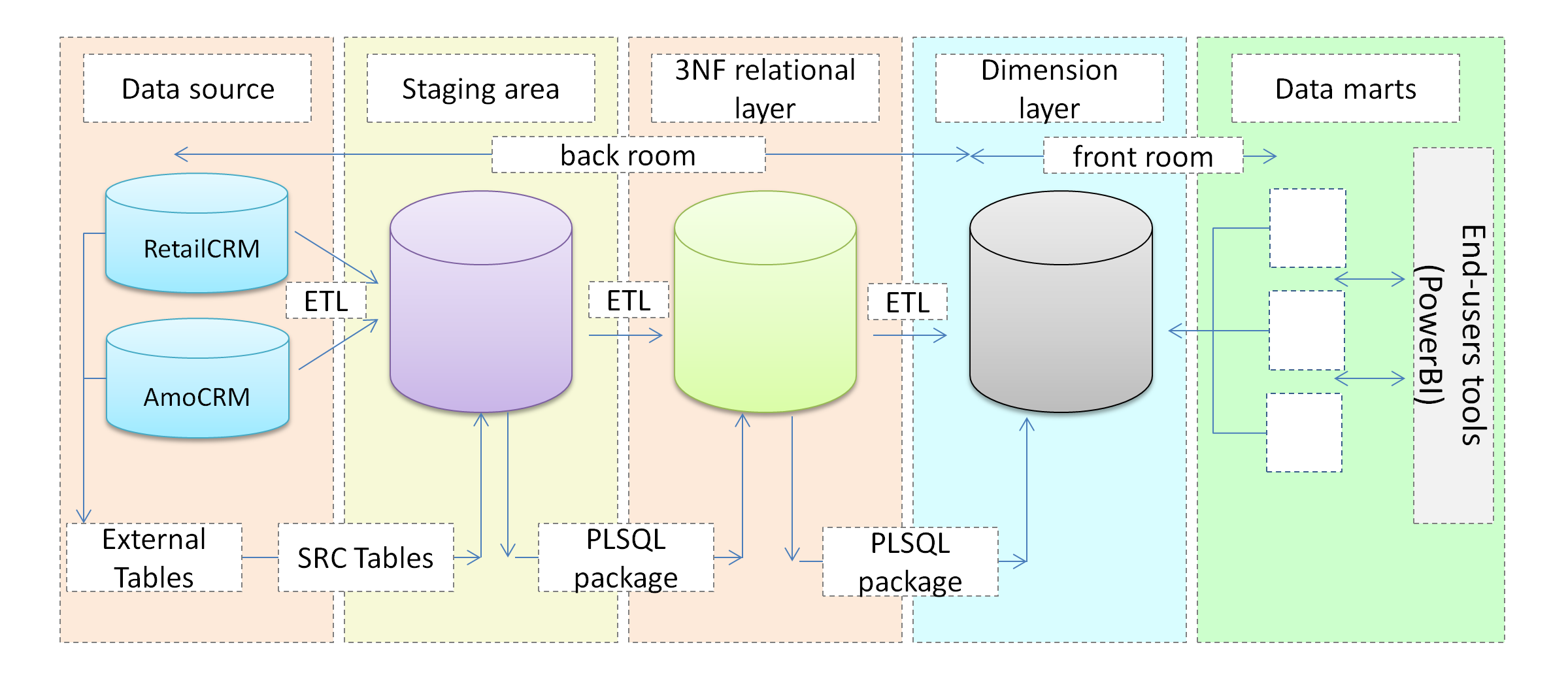
Step 1: Import Source Systems (RetailCRM and AmoCRM) in external tables.

Step 2: Staging area and source tables in Oracle DB are developed. ETL scripts are developed to load them with data from the source systems using external tables.

Step 3: All data was loaded and transformed to 3NF.

Step 4: Star layer based on 3NF was designed.

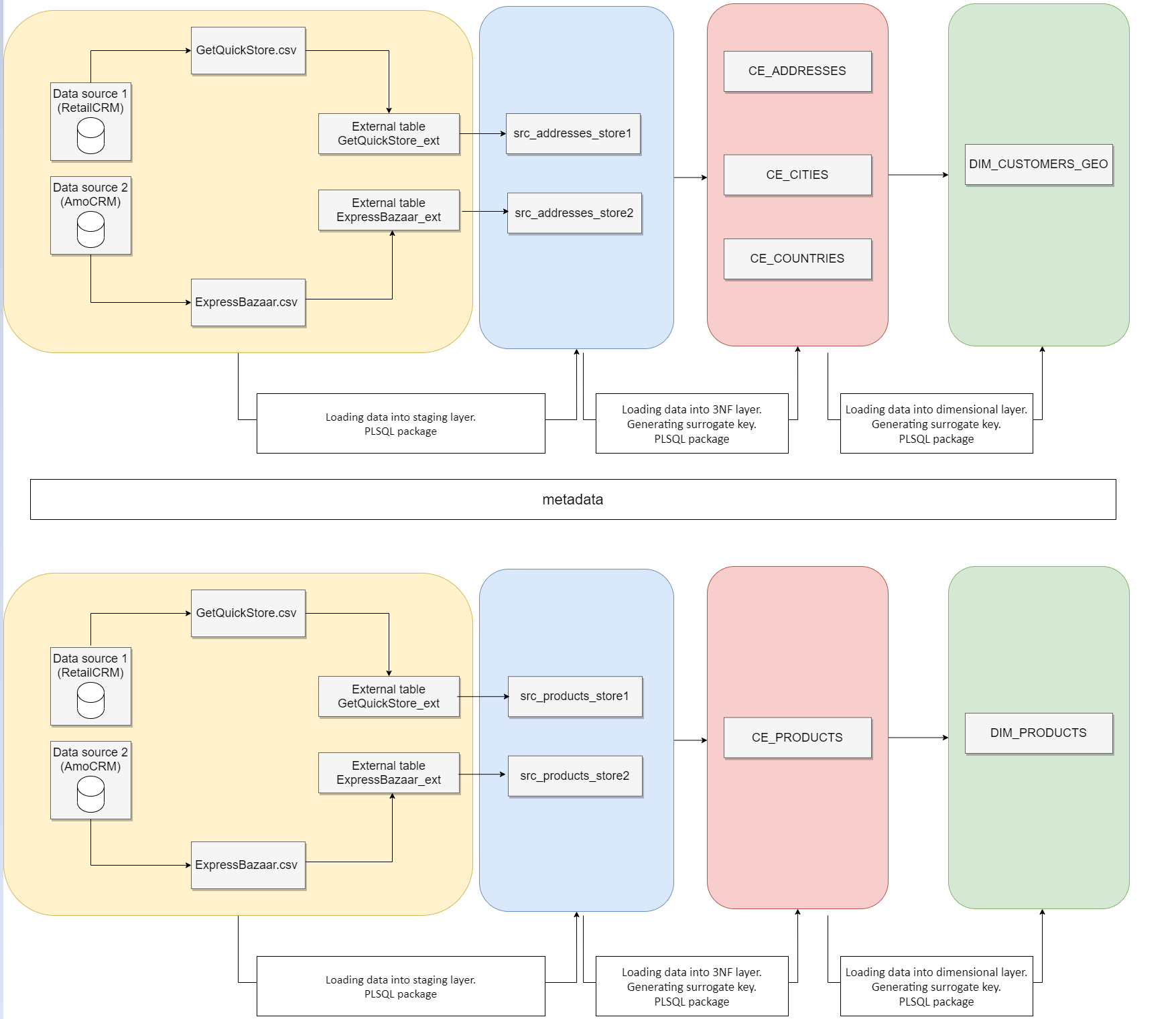
Step 5: Data marts are created from data warehouse and represented to end users by PowerBI.

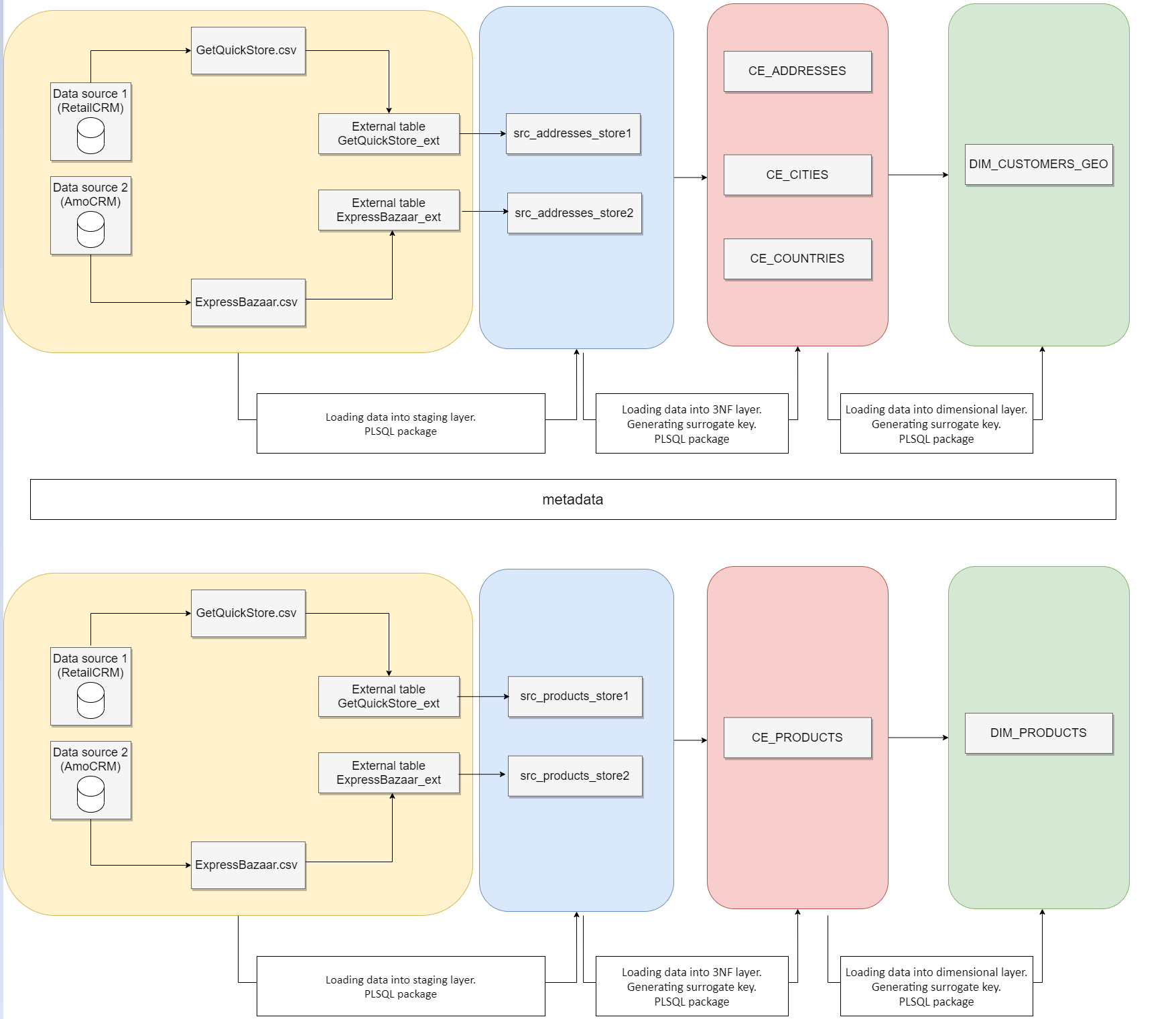


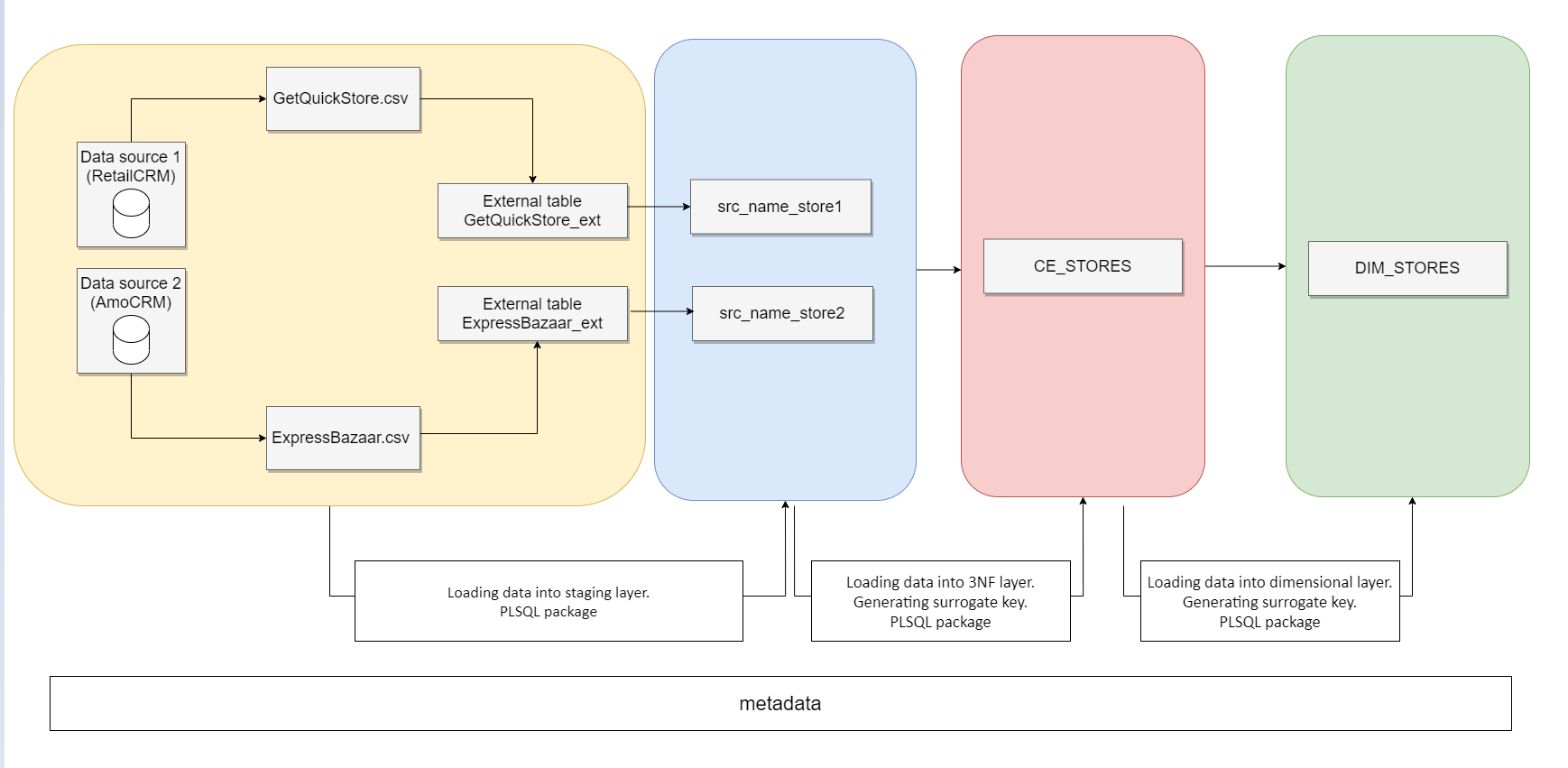
# Data Flow

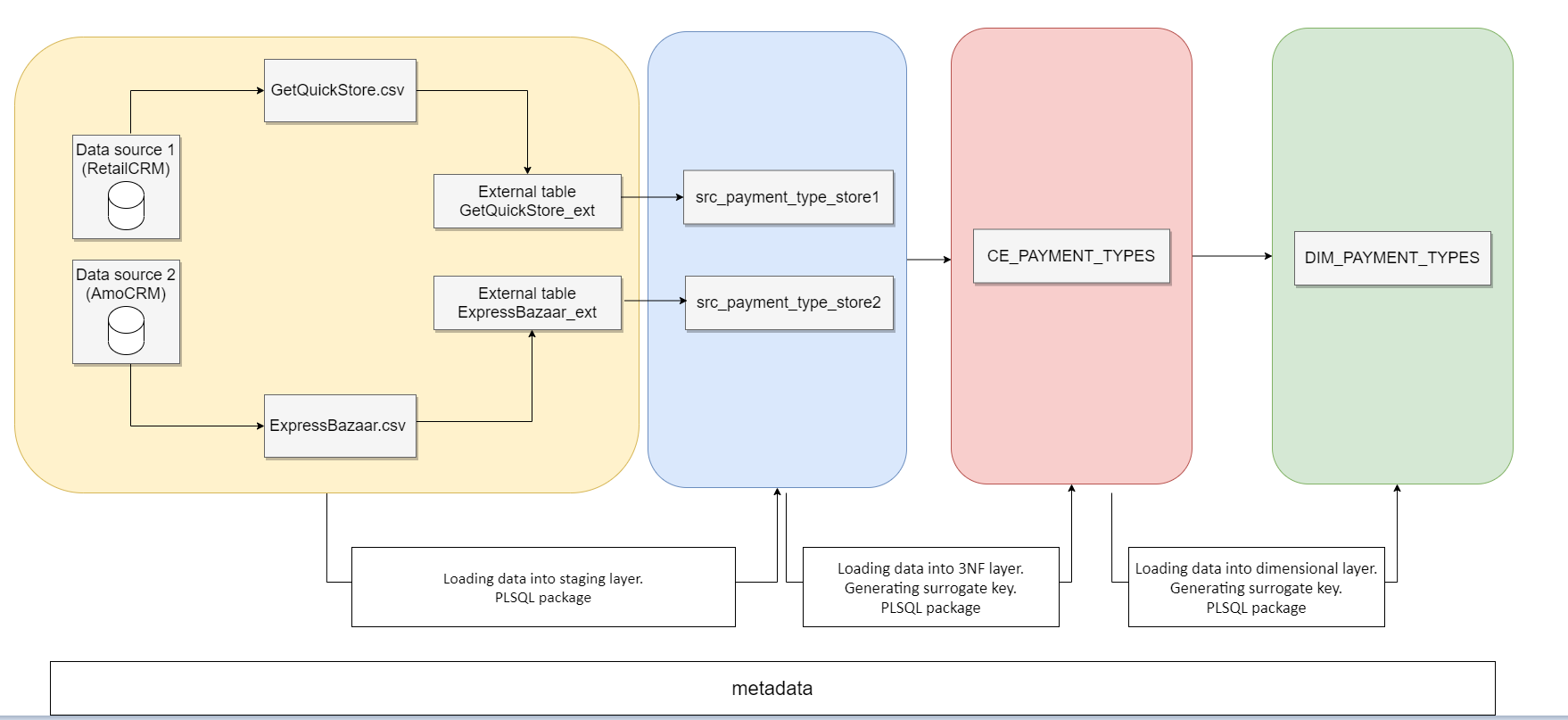
Data flow shows the transfer of information from one part of the system to another.

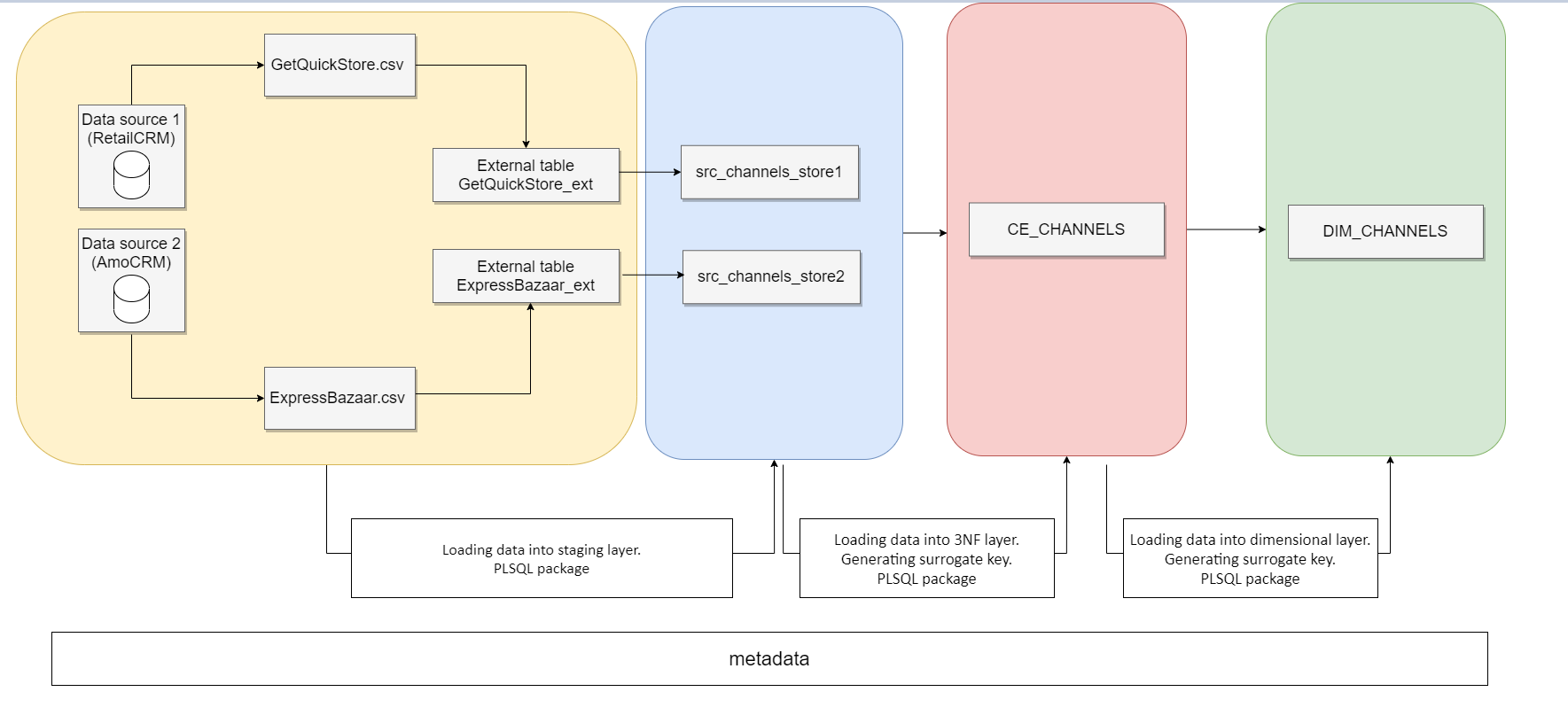
This diagrams was created using such tool as Draw.io.

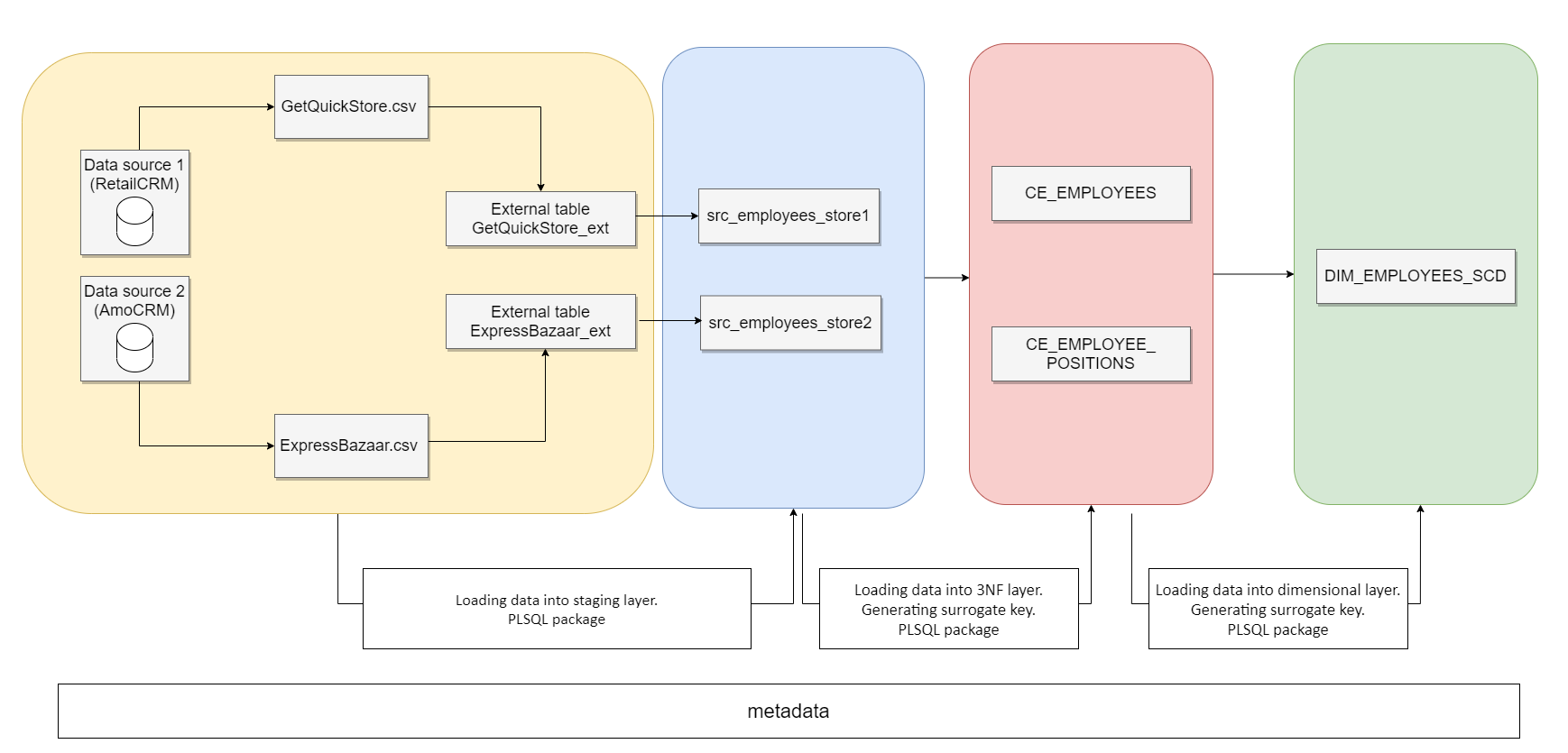


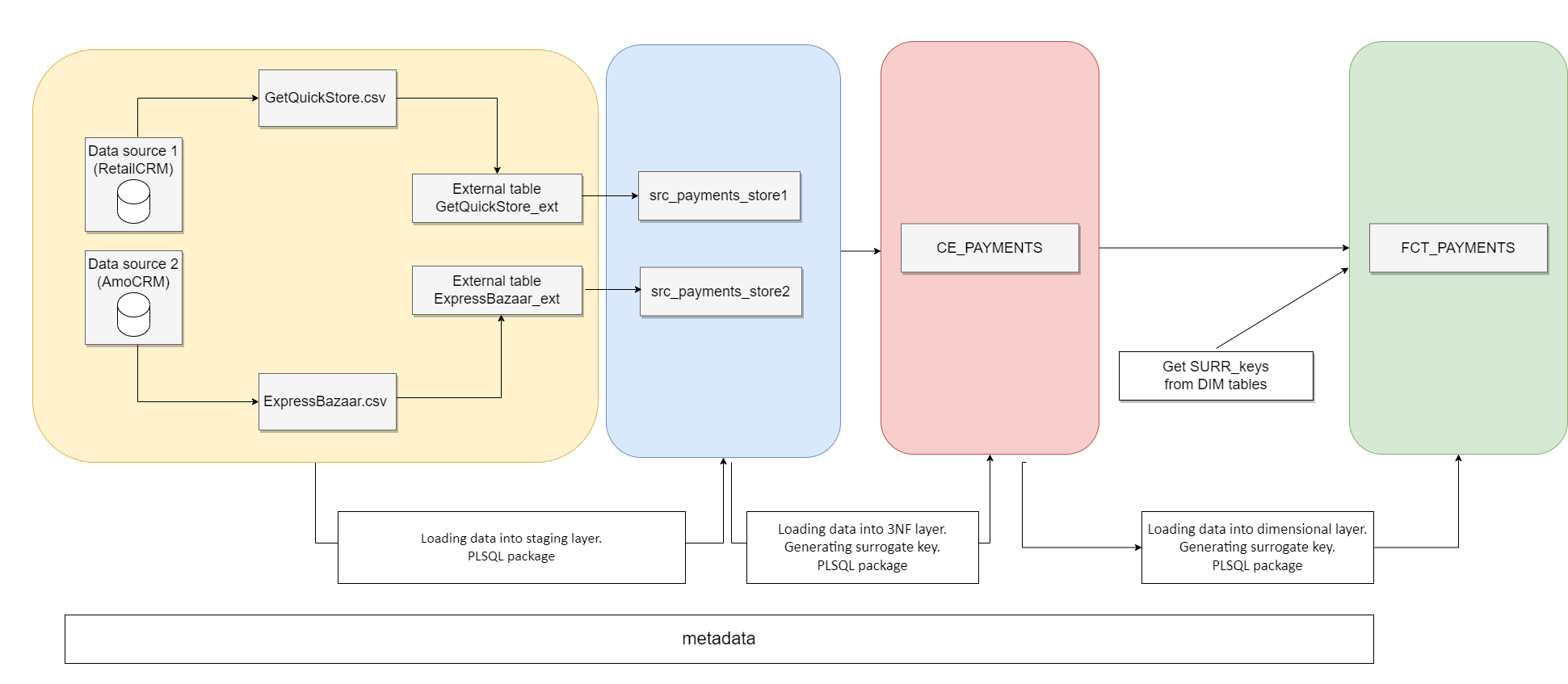












# PartiTIONING STRATEGY

Composite partitioning is a combination of the basic data distribution methods; a table is partitioned by one data distribution method and then each partition is further subdivided into subpartitions using a second data distribution method. All subpartitions for a given partition represent a logical subset of the data.

I think that it’s possible to use composite range-list partitioning.

Composite range-list partitioning is commonly used for large tables that store historical data and are commonly accessed on multiple dimensions. Often the historical view of the data is one access path, but certain business cases add another categorization to the access path. For example, regional account managers are very interested in how many new customers they signed up in their countries in a specific time period.

Range partitioning is often used to organize data by time intervals on a column of type DATE. Thus, most SQL statements accessing range partitions focus on time frames. An example of this is a SQL statement similar to "select data from a particular period in time". In such a scenario, if each partition represents data for one month, the query "find data of month DEC-2021" must access only the December partition of year 2021. This reduces the amount of data scanned to a fraction of the total data available, an optimization method called partition pruning.

We can use list partitioning when we want to specifically map rows to partitions based on discrete values. For example, all the customers from USA are stored in one partition and customers from other countries are stored in different partitions. Account managers who analyze their accounts by countries can take advantage of partition pruning.