Data Warehousing Guide - Following the Data Warehousing Process - 2/4

qimia.io/en/blog/Data-Warehousing-Guide-Following-the-Data-Warehousing-Process

Introduction

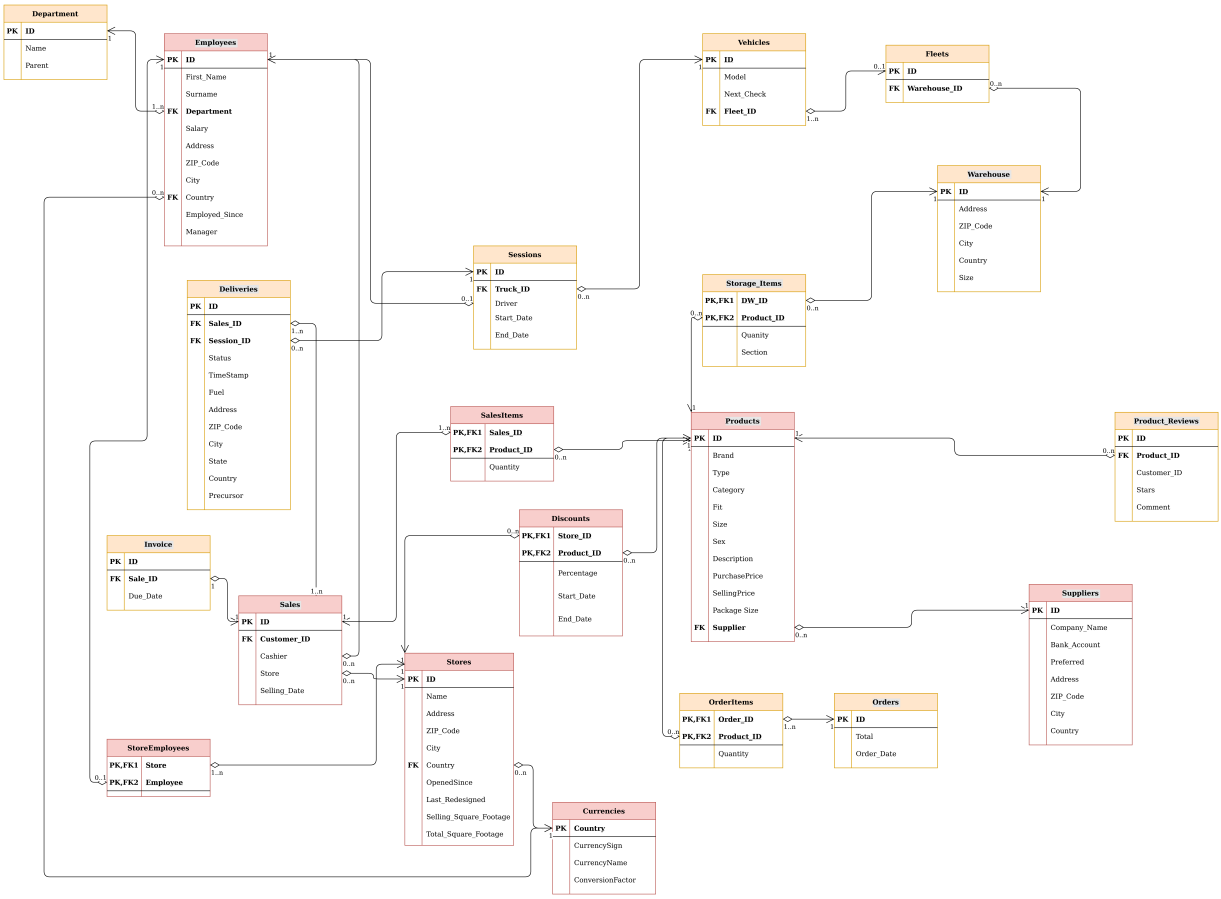
Welcome back to our tutorial series on data warehousing! Last time we presented the fundamentals of data warehousing, including fact tables, dimension tables, and management tools.

In this second part of the series, our focus will be on showing our existing OLTP schema, choosing fact and dimension tables, and defining the bus matrix.

1. OLTP Schema

In the vast majority of cases, the contents of a data warehouse will be sourced from a normalized SQL OLTP database. To facilitate a better understanding of the data warehousing process, we build our own OLTP schema resembling a casual operational database of a clothing retailer.

The graphic below shows the schema as defined by us. For our tutorial, we focus on the tables that are marked red. Be aware that the schema is by no means perfect since we made a few sacrifices to keep it clear and rather simple for the tutorial.



Retail OLTP Schema

We then built a generator to fill the tables with artificial data. You can find both the generator and the result files on our Github. Simply follow the repository instructions to set up the OLTP database.

2. Building the Bus Matrix

As described in our first article, Data Warehousing Guide - Core Concepts, building the star schema can be seen as a four-step process.

2.1. Select Business Process

Our OLTP schema includes multiple processes that are potential candidates for fact tables. We will demonstrate the warehousing process with the example of the sales process and the attributes that belong to it, like stores and employees.

2.2. Declare the Grain

When declaring the grain, we always want to have a fact table that has the most atomic grain, so that users are not restricted in the questions they can ask. We have three main candidates for the grain.

2.2.1. Grain: Per Sales Transaction

A sales transaction in our case is defined by a customer buying one or more products and paying for them in one go. Declaring the grain at this level will fill the fact table with one row per sales transaction. But we can go more atomic.

2.2.2. Grain: Per Product on a Sales Transaction

Choosing the grain to be on product level gives us one row per product that the customer bought in his sales transaction. The clear advantage is that we can link information about the product to the fact table. That would not be possible if our fact table is defined at the sales transaction level, because the sales transaction is composed of multiple products with distinct characteristics. However, we are not yet at the most atomic level. If a customer buys three packs of the same socks, we would still have one row for the socks in our case. So we could go even more atomic.

2.2.3. Grain: Per Item on a Sales Transaction

Compared to declaring the grain at the product level, defining it at the item level does not yield any benefits. It could be the case if we were tracking the items in our source system, but we do not.

Our target schema is limited by our source schema! Since our source schema does not track items, we will not be able to link the fact table to additional information about it. Thus, declaring the grain at the item level will not give us any advantages and will actually harm us, since it will blow up the number of rows in our fact table.

The right choice for us is clearly to define the grain at the product level.

2.3. Identifying the Dimensions

Usually, talking to the business is the right way to find appropriate dimensions for the business process. Since we do not have a business, we instead have a look at the source schema instead. Looking at the source schema yields the following dimension candidates:

- date
- products
- suppliers
- sales
- employees
- stores
- currencies

A date dimension is nearly always given since processes usually contain some information on when they took place. Since products and suppliers are dependent on one another, they should be joined together. The same holds for stores and currencies. It is a best practice to keep the dimensions completely independent from each other.

Do not be afraid to join tables together, even if it enlarges the dimension table. Keep in mind that we want to avoid building a snowflake schema because joins are expensive. When it comes to storage use, the size of the dimensions is usually negligible in comparison to the fact table.

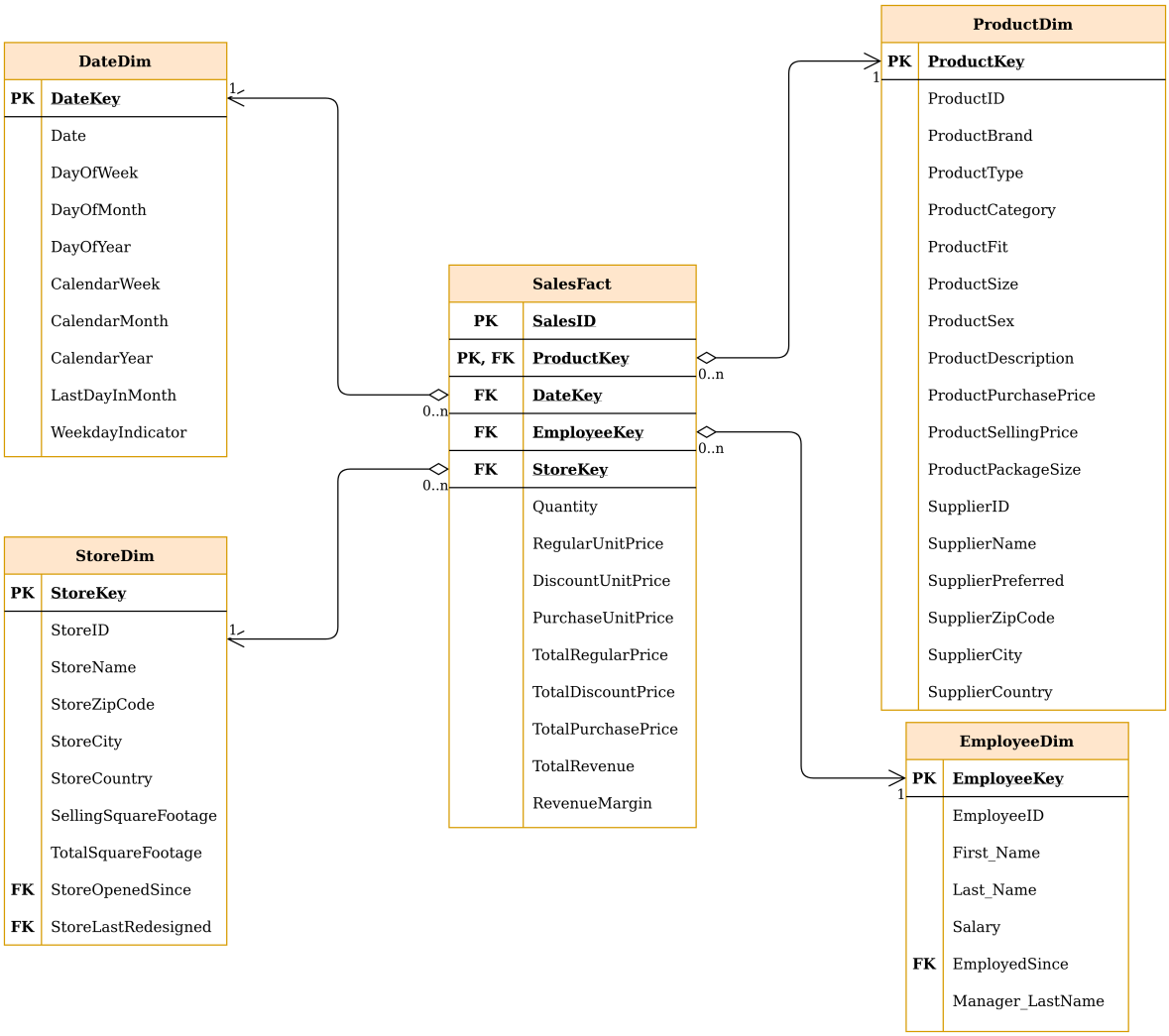
The sales table is nothing else than the parent or header to a child or line item, which is the sales items in our case. Since sales items are the basis for our fact table, all the attributes of sales are inherited to sales items. The sales dimension is left with nothing but its ID, which we subsequently also include in the fact table and avoid having another dimension table. The sales dimension is an example of a degenerate dimension.

2.4. Identifying the Facts

As facts, we can use the measurements that are taken during the process that we capture. We can also use attributes of other related tables that can be computed and joined into the fact table. In our case, we go for the following facts:

- quantity
- regular unit price
- discount unit price
- purchase unit price
- total regular price
- total discount price
- total purchase price
- total revenue
- revenue margin

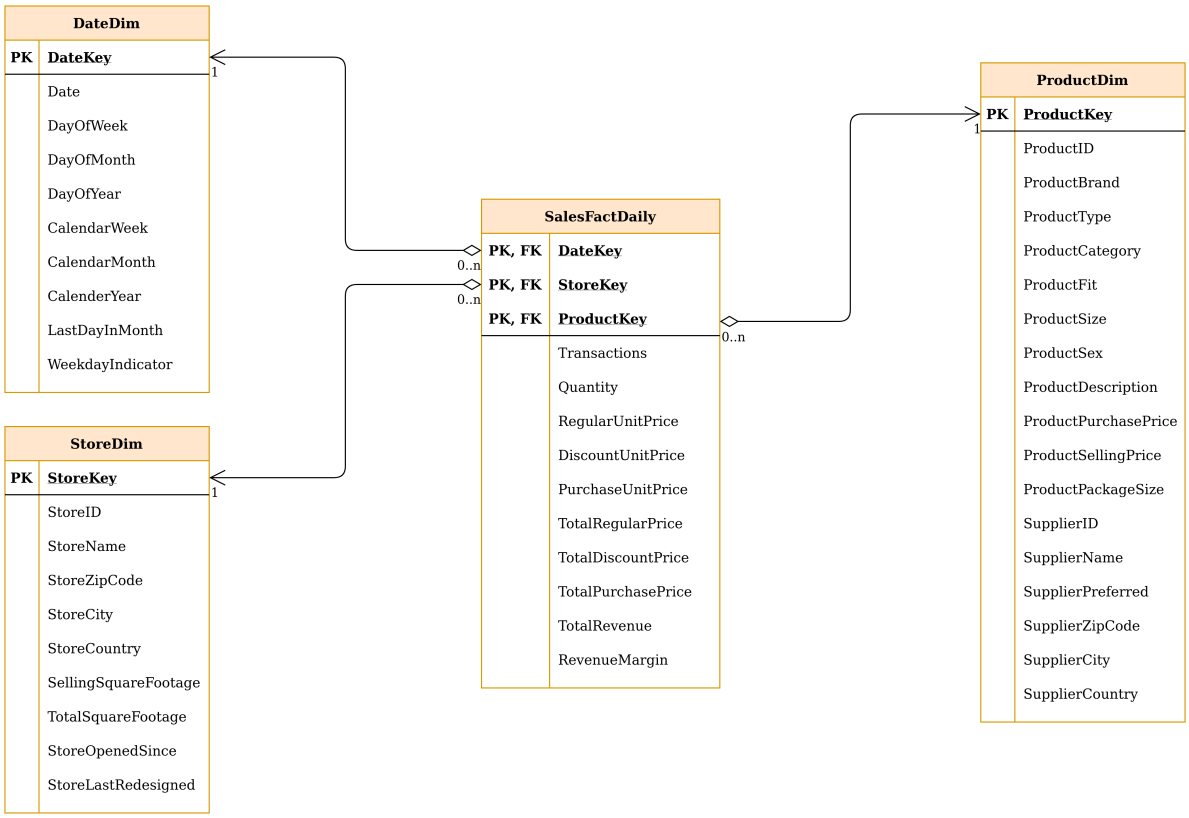
Our star schema for the atomic sales looks like this:



Atomic Star Schema

2.5. Adding Aggregated Fact Tables

Aggregated fact tables are not an absolute necessity, but can be added to contribute to better performance. In our case, we add a fact table for the daily sales, which can be seen as a *periodic snapshot table*. To produce the daily sales, we take the atomic fact table and group it by date, store, and product. We lose the employee dimension, but it can still be analyzed using the more atomic fact table.



Aggregated Star Schema

2.6. Bus Matrix

Following the procedure would give us the following bus matrix:

	Date	Product	Store	Employee	Fleet	...
Sales	X	X	X	X		...
Discounts	tbd	tbd	tbd	tbd	tbd	...
Deliveries	tbd	tbd	tbd	tbd	tbd	...
Purchases	tbd	tbd	tbd	tbd	tbd	...

Bus Matrix

Defining the dimensions for all the business processes leads to a matrix that will eventually enable the data engineering team to define a number of *conformed dimensions*. These are dimensions that are shared across multiple fact tables and enable

analysis that spans across the value chain. The single rows in the bus matrix should be taken care of one by one, incrementally integrating the new process and adding tangible value with each iteration.

Wrapping It Up

This time, we got to know the source schema that our fictional OLTP database is running on. We also learned how to build the star schemas and the bus matrix by following the four-step process.

Our next article will show you how we define the tables for the dimensions and sales facts. See you in part 3, Data Warehousing Guide - Defining Dimension and Fact Table.

For the next part of the tutorial click here