Unit 05 Notes

Dimensional Models

A **dimensional model** is a data model designed to be easy to understand and query by the business users, and offer good query response times to the same users. This model is easy to understand because it groups the data according to business categories. Business users can thus navigate the model and ignore categories that are not relevant to them. The model offers good query response time to business users by minimizing the number of table joins required to answer business queries.

Dimensional models stored in relational DBs are called **star schemas**. Dimensional models stored in multidimensional OLAP structures are called **cubes**. Note that both the dimensional models and the 3NF models are logical models of data that can be physically stored in relational databases. Thus, both models can be represented by **Entity-Relationship Diagrams** (**ERD)**, as illustrated in Figure 1.

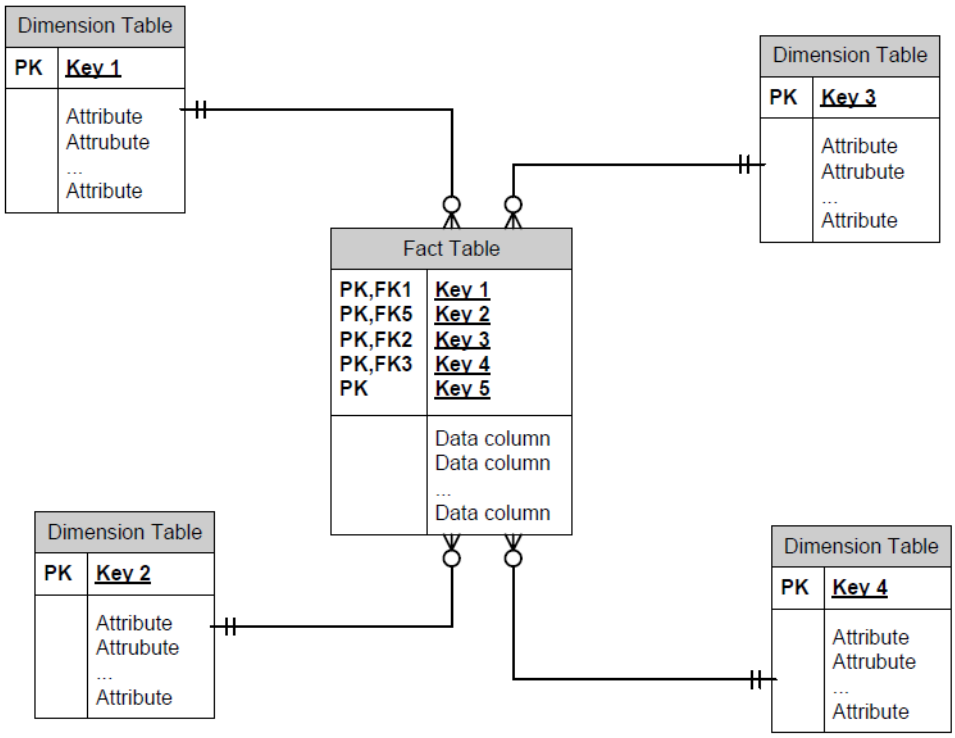

Figure 1 Dimensional Model represented by an ERD

Dimensional models are sometimes characterized as **de-normalized**. To be precise, fact tables are normalized to 3NF, and dimension tables typically resemble 2NF. This degree of de-normalization makes the data model easier to understand and query by non-IT users. In contrast, the 3NF model is designed to eliminate data redundancies and optimize transactions that update the tables. Since business users are running queries rather than transactions, and they are less concerned about redundancies, 3NF models may not be their best choice.

Dimensional modeling is the **logical design** method for building a dimensional model. Dimensional modeling divides the data into **facts** (measurements) and **dimensions** (context). Facts are usually numeric data values captured by operational data systems that support the organization's business processes. Dimensions are usually textual data values that represent the context in which the facts were measured and recorded in the data base. Dimensions describe who, when, where, why, and how facts were recorded. Each business process of an organization is represented by a **fact table**, consisting of facts, and a set of **dimension tables**, storing dimensions. Figure 2 presents an example.

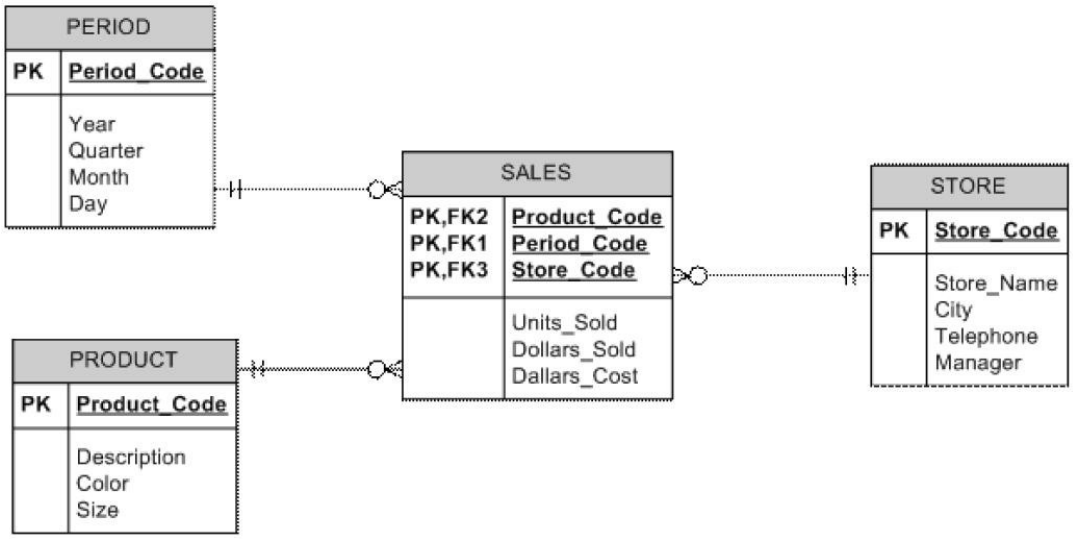

Figure 2 Facts (sales) and dimensions (periods, products, and stores)

The Fact Table

The fact table stores the numeric data that takes variable values (they are not known before being measured). Constant numeric values are dimensions rather than facts, and some values can be both. The most useful facts are **additive**, i.e., their values can be summed up over multiple rows. Some facts are **conformed**; they are allowed to have the same standardized name in different tables.

Fact tables store more than 90% of the data, thus they tend can have billions of rows. For efficiency, every fact is stored in a single fact table that is shared across the organization. A fact table has a multipart key made up of foreign keys from all dimension tables linked to that fact table. This means that every fact table expresses a **many-to-many relationship**. The fact table's foreign keys can never be null. The primary key of this table is typically a subset of its foreign keys.

The fact table should contain data at the lowest **granularity** captured by the business process. This allows users to ask precise questions and get precise answers. All measurement events are **transactions**, **periodic snapshots**, or **accumulating snapshots**. A transactional fact table has a row for each transaction; once stored in the fact table, a transaction is not changed. This is the lowest granularity and it has the largest number of dimensions associated with it. A periodic-snapshot fact table has a row for each periodic snapshot of the business process (e.g., a monthly loan); rows are not changed once

inserted. Most financial reports are such snapshots. Such tables complement rather than substitute for transactional tables. Accumulating-snapshot fact tables have one row for each snapshot, but update their rows in place as the measurement event progresses. These tables use two additional types of facts to manage the accumulating snapshots: milestone counters (which can be used to count rows that have reached a certain milestone in the workflow) and time intervals between milestones. These tables are less common and should be accompanied by transactional fact tables.

Dimension Tables

Unlike fact tables, which are filled with just keys and numeric measurements, dimension tables are filled with descriptive fields. These fields are used to constrain/filter queries, and to label query result sets. Dimensions correspond to the "by" words used by our business users in queries and reports (in Figure 2, these are “by” period, “by” product, and “by” store). Dimension attributes should be verbose (full words), descriptive, and complete (no missing values; replace Null values with "Unknown" or "N/A"). Codes and abbreviations stored in dimension tables should be accompanied by their decodes.

Dimension tables usually represent **many-to-one hierarchical relationships**. This is a consequence of de-normalization. For example, if products belong to stores, and stores belong to regions, we represent both hierarchies in a single dimension table for products, which has attributes for stores and regions. Creating separate dimensions for store and region decodes would save space but decrease model understandability and query performance. However, if we had 1000 products and 100 stores, we would create two dimensions rather than a single one with 100,000 product-stores.

In a dimension table with attributes that share the same values across many different rows, it sometimes makes sense to move these attributes into a separate table that is linked in the original dimension table by a foreign key. This is called an “**outrigger”** dimension (see an example in Figure 6-14, pp 267). However, this technique should be used sparingly; abusing it leads to dimension “**snowflaking”** (see Figure 6-13 on pp 266), which basically normalizes the dimension to 3NF. This decreases model understandability, the main purpose of using a dimensional model.

Two dimensions are called **conformed** if they are identical for all fact tables that they join to. Two dimensions are also called conformed if one of them (the less granular one) is a perfect subset of the other (the more granular one). **Standardized conformed dimensions** (or master dimensions) are essential in any data warehouse because they deliver consistency and facilitate the integration of business processes. They are the “bus” of the enterprise data warehouse, as they are shared across the enterprise, joining to multiple fact tables representing as many different business processes. It is critical that the development team use these dimensions in order to maximize the EDW benefits.

All dimension tables for a business process are linked to the same fact table, thus making a star-like structure called **star join**. The symmetry of this structure enables database engines to perform query optimization in a simple and predictable way. Dimension tables should have single-field keys. The best keys for them are **surrogate integer keys** that start at 1 and are incremented for every row.