**Star Schema and its components:**

The star schema is a data-modeling technique used to map multidimensional decision support data into a relational database. In effect, the star schema creates the near equivalent of a multidimensional database schema from the existing relational database. The star schema was developed because existing relational modeling techniques, ER, and normalization did not yield a database structure that served advanced data analysis requirements well.

The basic star schema has four components: facts, dimensions, attributes, and attribute hierarchies.

**I. Facts:**

Facts are numeric measurements (values) that represent a specific business aspect or activity. For example, sales figures are numeric measurements that represent product and/or service sales. Facts commonly used in business data analysis are units, costs, prices, and revenues. Facts are normally stored in a fact table that is the center of the star schema. The fact table contains facts that are linked through their dimensions.

Facts can also be computed or derived at run time. Such computed or derived facts are sometimes called metrics to differentiate them from stored facts. The fact table is updated periodically (daily, weekly, monthly, and so on) with data from operational databases.

**II. Dimensions:**

Dimensions are qualifying characteristics that provide additional perspectives to a given fact. Recall that dimensions are of interest because decision support data are almost always viewed in relation to other data. For instance, sales might be compared by product from region to region and from one time period to the next. Dimensions are normally stored in dimension tables. The following diagram depicts a star schema for sales with product, location, and time dimensions.

![Star Schema Diagram](image)

**III. Attributes:**

Each dimension table contains attributes. Attributes are often used to search, filter, or classify facts. Dimensions provide descriptive characteristics about the facts through their attributes. Therefore, the data warehouse designer must define common business attributes that will be used by the data analyst to narrow a search, group information, or describe dimensions. For example Region, state, city are dimensions of Location, Product type, product ID are dimensions of Product.

**IV. Attribute Hierarchies:**

Attributes within dimensions can be ordered in a well-defined attribute hierarchy. The attribute hierarchy provides a top-down data organization that is used for two main purposes: aggregation and drill-down/roll-up data analysis. For example, the following figure shows how the location dimension attributes can be organized in a hierarchy by region, state, city, and store.
The attribute hierarchy provides the capability to perform drill-down and roll-up searches in a data warehouse. For example, suppose a data analyst looks at the answers to the query: How does the 2009 month-to-date sales performance compare to the 2010 month-to-date sales performance? The data analyst spots a sharp sales decline for March 2010. The data analyst might decide to drill down inside the month of March to see how sales by regions compared to the previous year.

**Star Schema Representation:**

Facts and dimensions are normally represented by physical tables in the data warehouse database. The fact table is related to each dimension table in a many-to-one (M:1) relationship. In other words, many fact rows are related to each dimension row. Using the sales example, you can conclude that each product appears many times in the SALES fact table.

Fact and dimension tables are related by foreign keys and are subject to the familiar primary key/foreign key constraints. The primary key on the “1” side, the dimension table, is stored as part of the primary key on the “many” side, the fact table. Because the fact table is related to many dimension tables, the primary key of the fact table is a composite primary key.

The following figure illustrates the relationships among the sales fact table and the product, location, and time dimension tables. To show you how easily the star schema can be expanded, a customer dimension has been added to the mix. Adding the customer dimension merely required including the CUST_ID in the SALES fact table and adding the CUSTOMER table to the database.

The composite primary key for the SALES fact table is composed of TIME_ID, LOC_ID, CUST_ID, and PROD_ID. Each record in the SALES fact table is uniquely identified by the combination of values for each of the fact table’s foreign keys. By default, the fact table’s primary key is always formed by combining the foreign keys pointing to the dimension tables to which they are related.
Performance-Improving Techniques for the Star Schema:

The creation of a database that provides fast and accurate answers to data analysis queries is the data warehouse design’s prime objective. Therefore, performance-enhancement actions might target query speed through the facilitation of SQL code as well as through better semantic representation of business dimensions. These four techniques are often used to optimize data warehouse design:

- Normalizing dimensional tables.
- Maintaining multiple fact tables to represent different aggregation levels.
- Denormalizing fact tables.
- Partitioning and replicating tables.

1. **Normalizing Dimensional Tables:**
   Dimensional tables are normalized to achieve semantic simplicity and facilitate end-user navigation through the dimensions. For example, if the location dimension table contains transitive dependencies among region, state, and city, you can revise those relationships to the 3NF (third normal form).

2. **Maintaining Multiple Fact Tables That Represent Different Aggregation Levels:**
   You can also speed up query operations by creating and maintaining multiple fact tables related to each level of aggregation (region, state, and city) in the location dimension. These aggregate tables are precomputed at the data-loading phase rather than at run time. The purpose of this technique is to save processor cycles at run time, thereby speeding up data analysis. An end-user query tool optimized for decision analysis then properly accesses the summarized fact tables instead of computing the values by accessing a lower level of detail fact table.

3. **Denormalizing Fact Tables:**
   Denormalizing fact tables improves data access performance and saves data storage space. The latter objective, however, is becoming less of an issue. Data storage costs decrease almost daily, and DBMS limitations that restrict database and table size limits, record size limits, and the maximum number of records in a single table have far more negative effects than raw storage space costs. Denormalization improves performance by using a single record to store data that normally take many records.

4. **Partitioning and Replicating Tables:**
   Table partitioning and replication are particularly important when a BI system is implemented in dispersed geographic areas. Partitioning splits a table into subsets of rows or columns and places the subsets close to the client computer to improve data access time. Replication makes a copy of a table and places it in a different location, also to improve access time.