Design efficient database systems with schemas

You have been learning about how business intelligence professionals use data models and schemas to organize and optimize databases. As a refresher, a schema is a way of describing the way something is organized. Think about data schemas like blueprints of how a database is constructed. This is very useful when exploring a new dataset or designing a relational database. A database schema represents any kind of structure that is defined around the data. At the most basic level, it indicates which tables or relations make up the database, as well as the fields included on each table.

This reading will explain common schema types you might encounter on the job.

Types of schemas

Star and snowflake

You've already learned about the relational models of star and snowflake schemas. Star and snowflake schemas share some things in common, but they also have a few differences. For instance, although they both share dimension tables, in snowflake schemas, the dimension tables are normalized. This splits data into additional tables, which makes the schemas a bit more complex.

A **star schema** is a schema consisting of one or more fact tables referencing any number of dimension tables. As its name suggests, this schema is shaped like a star. This type of schema is ideal for high-scale information delivery and makes read output more efficient. It also classifies attributes into facts and descriptive dimension attributes (product ID, customer name, sale date).

Here's an example of a star schema:

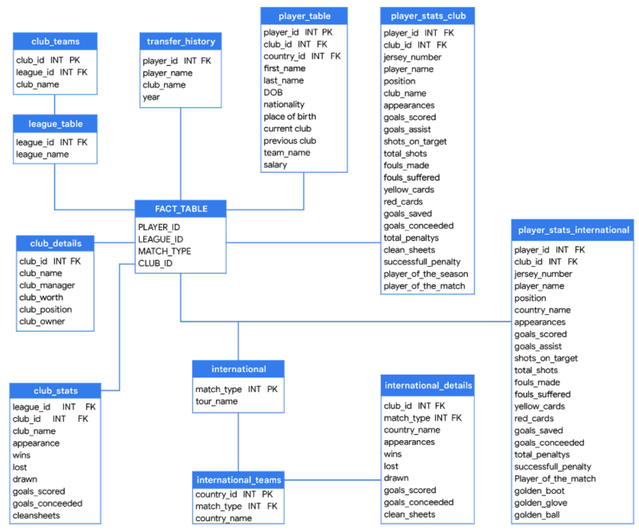
In this example, this company uses a star schema to keep track of sales information within their tables. This includes:

- Customer information
- Product information
- The time the sale is made
- Employee information

All the dimension tables link back to the sales_fact table at the center, which confirms this is a star schema.

A **snowflake schema** is an extension of a star schema with additional dimensions and, often, subdimensions. These dimensions and subdimensions create a snowflake pattern. Like snowflakes in nature, a snowflake schema—and the relationships within it—can be complex. Snowflake schemas are an organization type designed for lightning-fast data processing.

Below is an example of a snowflake schema:



Perhaps a data professional wants to design a snowflake schema that contains sports player/club information. Start at the center with the fact table, which contains:

- PLAYER_ID
- LEAGUE_ID
- MATCH_TYPE
- CLUB_ID

This fact table branches out to multiple dimension tables and even subdimensions. The dimension tables break out multiple details, such as player international and player club stats, transfer history, and more.

Flat model

Flattened schemas are extremely simple database systems with a single table in which each record is represented by a single row of data. The rows are separated by a delimiter, like a column, to indicate the separations between records. Flat models are not relational; they can't capture relationships between tables or data items. Because of this, flat models are more often used as a potential source within a data system to capture less complex data that doesn't need to be updated.

Here is a flat table of runners and times for a 100-meter race:

runner_name	time
Agnes Klaus	10.75
Emilia Frye	10.88
Isla Gomez	10.97
Jacinta Kiefer	11.02
Judith Lina	11.03
Milena Harris	10.99
Rachel Brittain	11
Tina Sato	10.77
Torry Kready	11.05
Whitney Perez	12

This data isn't going to change because the race has already occurred. And, it's so simple, it's not really worth the effort of integrating it into a complex relational database when a simple flat model suffices.

As a BI professional, you may encounter flat models in data sources that you want to integrate into your own systems. Recognizing that these aren't already relational models is useful when considering how best to incorporate the data into your target tables.

Semi-structured schemas

In addition to traditional, relational schemas, there are also semi-structured database schemas which have much more flexible rules, but still maintain some organization. Because these databases have less rigid organizational rules, they are extremely flexible and are designed to quickly access data.

There are four common semi-structured schemas:

Document schemas store data as documents, similar to JSON files. These documents store pairs of fields and values of different data types.

Key-value schemas pair a string with some relationship to the data, like a filename or a URL, which is then used as a key. This key is connected to the data, which is stored in a single collection. Users directly request data by using the key to retrieve it.

Wide-column schemas use flexible, scalable tables. Each row contains a key and related columns stored in a wide format.

Graph schemas store data items in collections called nodes. These nodes are connected by edges, which store information about how the nodes are related. However, unlike relational databases, these relationships change as new data is introduced into the nodes.

Conclusion

As a BI professional, you will often work with data that has been organized and stored in different ways. Different database models and schemas are useful for different things, and knowing that will help you design an efficient database system!

Mark as completed