Relational Databases and SQL

Krish Ganotra

January 2021

1 Introduction

We live in an age of information where one of the most powerful tools available is having large amounts of data. Many major corporations including Google, Facebook, and Amazon rely their entire business on the ability to gather and use data effectively. However, with complicated data that has numerous relationships and connections, managing all this data can get complicated. Relational Databases are the current state of the art for storing this data efficiently to preserve the complex relationships while saving on storage and making the data simple to access and analyze.

2 How Do Relational Databases Work?

A relational database organizes data into tables which can be linked—or related—based on data common to each. This capability enables you to retrieve an entirely new table from data in one or more tables with a single query. It also allows you and your business to better understand the relationships among all available data and gain new insights for making better decisions or identifying new opportunities.

For example, imagine your company maintains a customer table that contains company data about each customer account and one or more transaction tables that contain data describing individual transactions.

The columns (or fields) for the customer table might be *Customer ID, Company Name, Company Address*, etc.; the columns for a transaction table might be *Transaction Date, Customer ID, Transaction Amount, Payment Method*, etc. The tables can be related based on the common Customer ID field. You can, therefore, query the table to produce valuable reports, such as a consolidated customer statement.

Report generators take these queries and run them on demand to create formal reports. Many of the documents businesses run to track inventory, sales, finance, or even perform financial projections come from a relational database operating behind the scenes.

3 What is SQL?

You can communicate with relational databases using Structured Query Language (SQL), the standard language for interacting with management systems. SQL allows the joining of tables using a few lines of code, with a structure most nontechnical employees can learn quickly.

With SQL, analysts do not need to know where the order table resides on disk, how to perform the lookup to find a specific order, or how to connect the order and customer tables. The database compiles the query and figures out the correct data points.

4 Benefits

The primary benefit of the relational database approach is the ability to create meaningful information by joining the tables. Joining tables allows you to understand the relationships between the data, or how the tables connect. SQL includes the ability to count, add, group, and also combine queries. SQL can perform basic math and subtotal functions and logical transformations. Analysts can order the results by date, name, or any column.

Those features make the relational approach the single most popular query tool in business today.

Relational databases have several advantages compared to other database formats:

4.1 Flexibility

SQL has its a built-in language for creating tables called Data Definition Language (DDL). DDL allows you to add new columns, add new tables, rename relations, and make other changes even while the database is running and while queries are happening. This allows you to change the schema or how you model data on the fly.

4.2 Reduced redundancy

Relational databases eliminate data redundancy. The information for a single customer appears in one place—a single entry in the customer table. The order table only needs to store a link to the customer table. The practice of separating the data to avoid redundancy is called normalization. Progressional database designers make sure the tables normalize during the design process.

4.3 Ease of backup and disaster recovery

Relational databases are transactional—they guarantee the state of the entire system is consistent at any moment. Most relational databases offer easy export and import options, making backup and restore trivial. These exports can happen even while the database is running, making restore on failure easy.

Modern, cloud-based relational databases can do continuous mirroring, making the loss of data on restore measured in seconds or less. Most cloud-managed services allow you to create Read Replicas. These Read Replicas enable you to store a read-only copy of your data in a cloud data center. Replicas can be promoted to Read/Write instances for disaster recovery as well.

5 Types of Relational Databases

Many database products implement the relational database model. They either use a SQL database for processing or can at least process SQL statements for requests and database updates. These databases range from small, desktop systems to massive cloud-based systems. They can be either open source and internally supported, open source with commercial support, or commercial closed-source systems. Here are a few of the more popular ones:

5.1 MySQL

MySQL is a common and easy to start a low-memory/disk/CPU database. It supports all the basic SQL commands, along with transactions and Atomicity, Consistency, Isolation, Durability (ACID) performance. MySQL is the most common database integrated with WordPress sites.

5.2 PostgreSQL

PostgreSQL is also open source. It provides enterprise features such as security, scalability, and support for more automation through a command-line interface, as well as direct access over the web. PostgreSQL supports stored procedures, which is a more complex programming language built on top of SQL. Teams can use stored procedures to do data extraction, transform, and load between systems. Examples of this use case include claims processing for insurance processing and order processing for complex orders. Postgres also works with qGIS or Geo Server to store and save global information.

6 Database schemes

A schema of a database is the structure defined in a formal language. In SQL, there are two formal languages: Data Definition Language (DDL) and Data Manipulation Language (DML).

To select data from a table, update it, or delete it, programmers use DML. To work on the database and to create and define the tables and relations, they use DDL or a tool to create the DDL.

DDL includes commands like CREATE, DROP, ALTER, COMMENT, and RENAME. The first DDL command is CREATE TABLE. This command defines the primary key (which must be unique), the fields, how those fields store data, and any other constraints on the table. Once defined, you can establish a

database as a combination of its definition language and the DML to INSERT the rows into the tables.

Here's an example of a DDL command to create an employee table. As you'll see, a schema can include what columns are required, which must be unique (a key), and which must have a reference in other tables.

In this example, the jobs table describes every job title, description, and job level. A separate table, the pay_grades table, shows the salary for each job level. Integers are whole numbers; char(100) reserves up to one hundred characters for text.

```
CREATE TABLE employees (
    employeeID integer UNIQUE NOT NULL,
    first_name char(100) NOT NULL,
    last_name char(100) NOT NULL,
    jobID int,
    birthdate date NOT NULL,
    governmentID char(9),
    FOREIGN KEY(jobID) REFERENCES jobs(jobID)
);
```

What happens when a row is deleted from the jobs table? DDL allows the programmer to specify what to do in the case. The column in employees could become automatically updated to NULL, as it will in this case, or it could be set to casecade delete. For example, if a branch library is deleted due to a closure, the books they have not transferred out might be deleted as well.

Some relational databases offer the potential to have multiple schemas, with different tables, along with the ability to grant read and write permissions to them. Multiple schemas mean an HR user might have HR information for employees but not confidential employee information (and vice versa), all in the same database.

7 Hosting SQL Servers

All of the data we put into our relational database must be stored somewhere, so we need storage. We also need to run SQL queries to add, delete, access, and modify data, so we need computer power. This seems like the perfect job for cloud computing. It is possible to host SQL servers on your own computers, but it is much more advantageous to do this in the cloud.

7.1 Benefits of Using Cloud Computing

- **Installation of Database Management Services** – Just like you need to install Python or Java to run code of those respective languages, you need to install a database management service that manages the database and interprets SQL command. These include MySQL and PostgreSQL. When using a SQL server from a public cloud provider, these services

come pre-installed and the cloud provider manages keeping the software up to date. Furthermore, you can have multiple servers running different services (one with MySQL and another with PostgreSQL), all managed in the cloud

- **Scalability** – Once you have large amounts of data (which is what you want), it becomes difficult to manage. Adding more storage onto a local machine is difficult, but cloud servers can automatically attach more storage so that you never have to worry about running out of storage space since you have too much date. Furthermore, large datasets require a ton of computer power to query, which your local machine can not and should not handle. This is also scaled in the cloud, with the option of attaching multiple CPUs to further increase the speed of labor-intensive queries.

- **Cloud Ecosystem** – Many public cloud providers provide integration between their many services. There are many available services made to handle data, from cleansing data, to gaining analytics, to running ML using the data, to changing the data format. With your data already stored in the cloud, you will have easy access to these numerous services to get the most out of your stored data