# What is a Database?

Learn about databases and how they can be used when building applications.

Share

Databases are an extremely important part of web application development. They are responsible for persisting organized collections of data for applications. This ability to store and access data for use at a later date allows applications to provide persistent experiences for their users and can be seen in use in almost any application in some form. Databases vary in their implementations and, as such, provide different advantages and disadvantages. These differences should be taken into account when designing an application as they can have a significant impact on overall performance.

## Relational Database

A *relational database*, commonly referred to as a SQL database, is a type of database that uses a structure that allows us to identify and access data in relation to another piece of data in the database. Often, data in a relational database is organized into tables. Data in a relational database table is stored as rows, called records, with each consisting of one or multiple columns that have a specific data type (ie: `INTEGER`, `VARCHAR`, `DATETIME`, etc). Relational databases come with many advantages, some of which include high accuracy and flexibility. Due to the normalized state of relational data it is only stored once within the database. This means that data only has to be updated in one location. Additionally, this structure allows for extreme query flexibility with users being able to `JOIN` data from multiple tables to create the specific, tailored dataset they are looking for at any given time.

While these advantages are great, they are mirrored by some disadvantages. One disadvantage is that normalization can lead to slower access times for queries that are extremely complex (pulling large amounts of data from multiple tables). Another disadvantage is cost. Relational databases are vertically scalable, which means that increasing load requires increasing server hardware power.

## Non-Relational Database

A *non-relational* database, commonly referred to as a NoSQL database, is any database that does not follow the relational model provided by traditional relational database management systems. Non-relational databases specialize in storing unstructured data that doesn't fit neatly into rows and columns.

Non-relational databases are great when scalability is key. The flexibility of their schema allows for updates to the database to handle changing application requirements fairly easily. Additionally, increased access load is handled horizontally rather than vertically as with their relational counterparts. This helps keep cost down as many cheaper commodity servers can be used rather than investing in expensive hardware. These two advantages combine to make non-relational databases prime candidates for Big Data demands.

While non-relational databases have great performance and cost at scale, they do suffer from some disadvantages. Due to the fact that the data is largely unstructured, accuracy can be more complex to maintain. Query flexibility is also somewhat degraded when it comes to accessing data. In order to service its high scalability needs, access patterns for non-relational data oftentimes have to be largely planned up front. While the ultimate outcome of this is extremely fast data access, it can limit the types of access, making subsets of queries extremely fast while leaving other queries that weren't pre-planned to suffer from potential performance penalties.

Graph Database

A *graph* database is a database that is designed to treat the relationships between data as equally important to the data itself. This means that data is not stored with a predefined model and is instead stored showing how each individual entity connects with others. Graph databases are extremely flexible as their schemas are able to adapt over time, adding new relationships between entities easily. This allows for quick adaptation to changing business needs without incurring too much technical debt. Additionally, graph databases offer very fast data lookup capabilities with the ability to have constant time traversals for large datasets against single entities. An area where graph databases can falter somewhat are in business intelligence and data analytics applications where aggregations are required. While this isn't an impossible task for these databases, it isn't fully optimized for this type of access.