

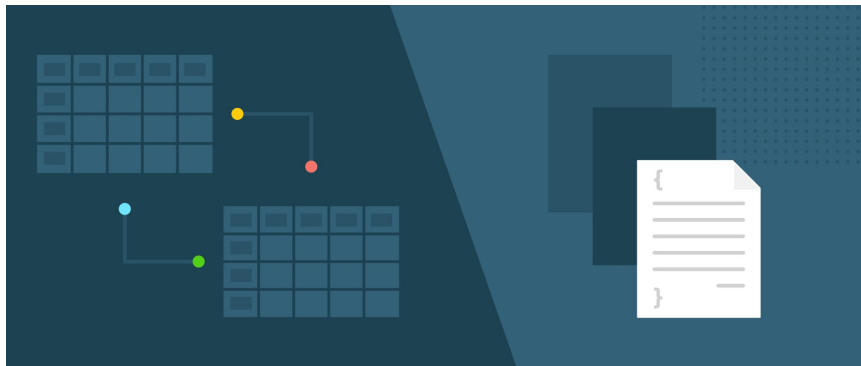


Upwork is the leading online talent solution. [Learn More](#)



[Hiring](#)

SQL vs. NoSQL Databases: What's the Difference?

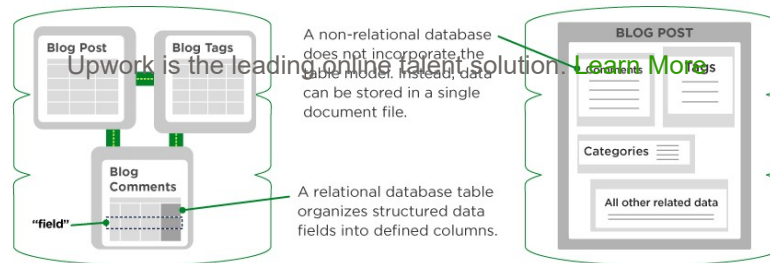


Carey Wodehouse

June 20, 2019 • 18 Min Read

In the world of [database technology](#), there are two main types of databases: [SQL](#) and [NoSQL](#)—or, relational databases and non-relational databases. The difference speaks to how they're built, the type of information they store, and how they store it. Relational databases are structured, like phone books that store phone numbers and addresses. Non-relational databases are document-oriented and distributed, like file folders that hold everything from a person's address and phone number to their Facebook likes and online shopping preferences.

We call them SQL and NoSQL, referring to whether or not they're written solely in structured query language (SQL). In this article, we'll explore what SQL is, how it makes these databases different, and how each type structures the data it holds so you can easily determine which type is right for you.



SQL: RELATIONAL DATABASES

First, let's take a look at one of the main features that separates these two systems: the way they structure data. A **relational database**—or, an SQL database, named for the language it's written in, Structured Query Language (SQL)—is the more rigid, structured way of storing data, like a phone book. Developed by IBM in the 1970s, a relational database consists of two or more tables with columns and rows. Each row represents an entry, and each column sorts a very specific type of information, like a name, address, and phone number. The relationship between tables and field types is called a **schema**. In a relational database, the schema must be clearly defined before any information can be added.

For a relational database to be effective, the data you're storing in it has to be structured in a very organized way. A well-designed schema minimizes data redundancy and prevents tables from becoming out-of-sync, a critical feature for many businesses, especially those that record financial transactions. A poorly designed schema can result in organizational headaches due to its rigidity. For example, a column designed to store U.S. phone numbers might require 10 digits because that's the standard for phone numbers in the U.S. This has the advantage of rejecting any invalid values (for example, if a number is missing an area code). However, if you need to change the schema (for instance, if you need to include an international phone number entry with more than 10 digits), then the entire database needs to be edited. Key takeaway: excellent organization results in a compromise in flexibility with a relational database.


Structured Query Language (SQL) is a programming language used by **database architects** to design relational databases. In an SQL database like MySQL, Sybase, Oracle, or IBM DB2, SQL executes queries, retrieves data, and edits data by updating, deleting, or creating new records. SQL is a lightweight, declarative language that does a lot of heavy lifting for the relational database, acting like a database's version of a server-side script. One particular advantage of SQL is its simple-yet-powerful JOIN clause, which allows developers to retrieve related data stored across multiple tables with a single command.

Another reason SQL databases remain popular is that they fit naturally into many venerable software stacks, including LAMP and Ruby-based stacks. These databases are well understood and widely supported, which can be a major advantage if you run into problems. [Learn More](#)



Popular SQL databases and RDBMS's

- **MySQL**—the most popular open-source database, excellent for CMS sites and blogs.
- **Oracle**—an object-relational DBMS written in the C++ language. If you have the budget, this is a full-service option with great customer service and reliability. Oracle has also released an Oracle NoSQL database.
- **IMB DB2**—a family of database server products from IBM that are built to handle advanced “big data” analytics.
- **Sybase**—a relational model database server product for businesses primarily used on the Unix OS, which was the first enterprise-level DBMS for Linux.
- **MS SQL Server**—a Microsoft-developed RDBMS for enterprise-level databases that supports both SQL and NoSQL architectures.
- **Microsoft Azure**—a cloud computing platform that supports any operating system, and lets you store, compute, and scale data in one place. A recent survey even put it ahead of Amazon Web Services and Google Cloud Storage for corporate data storage.
- **MariaDB**—an enhanced, drop-in version of MySQL.
- **PostgreSQL**—an enterprise-level, object-relational DBMS that uses procedural languages like [Perl](#) and Python, in addition to SQL-level code.



**Find the right freelancer
with a great job post**

[Read Now](#)

upwork

NOSQL DATABASES: NON- RELATIONAL & DISTRIBUTED DATA

If your data requirements aren't clear at the outset or if you're dealing with massive amounts of unstructured data,

you may not have the luxury of developing a relational database with clearly defined schema. Enter non-relational databases, which offer much greater flexibility than their traditional counterparts. Think of non-relational databases more like file folders, assembling related information of all

types. If a [WordPress](#) blog used a NoSQL database, each file could store data for a blog post: social likes, photos, text, metrics, links, and more.



Unstructured data from the web can include sensor data, social sharing, personal settings, photos, location-based information, online activity, usage metrics, and more. Trying to store, process, and analyze all of this unstructured data led to the development of schema-less alternatives to SQL. Taken together, these alternatives are referred to as NoSQL, meaning “Not only SQL.” While the term NoSQL encompasses a broad range of alternatives to relational databases, what they have in common is that they allow you to treat data more flexibly.

How do NoSQL databases work? Instead of tables, NoSQL databases are **document-oriented**. This way, non-structured data (such as articles, photos, social media data, videos, or content within a blog post) can be stored in a single document that can be easily found but isn’t necessarily categorized into fields like a relational database does. It’s more intuitive, but note that storing data in bulk like this requires extra processing effort and more storage than highly organized SQL data. That’s why [Hadoop, an open-source computing and data analysis platform](#) capable of processing huge amounts of data in the cloud, is so popular in conjunction with NoSQL database stacks.

NoSQL databases offer another major advantage, particularly to app developers: ease of access. Relational databases have a fraught relationship with applications written in object-oriented programming languages like Java, PHP, and Python. NoSQL databases are often able to sidestep this problem through APIs, which allow developers to execute queries without having to learn SQL or understand the underlying architecture of their database system.

COMMON TYPES OF NOSQL DATABASES

TYPES OF NON-RELATIONAL DATABASES



T Y P E S	PERFORMANCE	SCALABILITY	FLEXIBILITY	COMPLEXITY
KEY-VALUE STORE	high	high	high	none
COLUMN STORE	high	high	moderate	low
DOCUMENT	high	variable (high)	high	low
GRAPH DATABASE	variable	variable	high	high

1. **Key-value model**—the least complex NoSQL option, which stores data in a schema-less way that consists of indexed keys and values. *Examples: Cassandra, Azure, LevelDB, and Riak.*
2. **Column store**—or, wide-column store, which stores data tables as columns rather than rows. It’s more than just an inverted table—

sectioning out columns allows for excellent scalability and high performance. *Examples: HBase, BigTable, HyperTable.*



3. **Document database**—taking the key-value concept and adding more complexity, each document in this type of database has its own data, and its own unique key, which is used to retrieve it. It's a great option for storing, retrieving and managing data that's document-oriented but still somewhat structured. *Examples: MongoDB, CouchDB.*
4. **Graph database**—have data that's interconnected and best represented as a graph? This method is capable of lots of complexity. *Examples: Polyglot, Neo4J.*



Popular NoSQL Databases

- **MongoDB**—the most popular NoSQL system, especially among startups. A document-oriented database with JSON-like documents in dynamic schemas instead of relational tables that's used on the back end of sites like Craigslist, eBay, Foursquare. It's open-source, so it's free, with good customer service. Read more in [Should You Use MongoDB? A Look at the Leading NoSQL Database.](#)
- **Apache's CouchDB**—a true DB for the web, it uses the JSON data exchange format to store its documents; JavaScript for indexing, combining and transforming documents; and, HTTP for its API.
- **HBase**—another Apache project, developed as a part of Hadoop, this open-source, non-relational “column store” NoSQL DB is written in Java, and provides BigTable-like capabilities.
- **Oracle NoSQL**—Oracle's entry into the NoSQL category.
- **Apache's Cassandra DB**—born at Facebook, Cassandra is a distributed database that's great at handling massive amounts of structured data. Anticipate a growing application? Cassandra is excellent at scaling up. Examples: Instagram, Comcast, Apple, and Spotify.
- **Riak**—an open-source key-value store database written in Erlang. It has fault-tolerance replication and automatic data distribution built in for excellent performance.

What database solution is right for you?

Reasons to use a SQL database

When it comes to database technology, there's no one-size-fits-all solution. That's why many businesses rely on both relational and nonrelational databases for different tasks. Even as NoSQL databases gain popularity for their speed and scalability, there are still situations where a highly structured SQL database may be preferable. Here are a few reasons you might choose an SQL database:

1. **You need to ensure ACID compliancy (Atomicity, Consistency, Isolation, Durability).** ACID compliancy reduces anomalies and

protects the integrity of your database by prescribing exactly how transactions interact with the database. Generally, NoSQL databases sacrifice ACID compliancy for flexibility and processing speed, but for many e-commerce and financial applications, an ACID-compliant database remains the preferred option.

2. **Your data is structured and unchanging.** If your business is not experiencing massive growth that would require more servers and you're only working with data that's consistent, then there may be no reason to use a system designed to support a variety of data types and high traffic volume.

Reasons to use a NoSQL database

When all of the other components of your server-side application are designed to be fast and seamless, NoSQL databases prevent data from being the bottleneck. Big data is the real NoSQL motivator here, doing things that traditional relational databases cannot. It's driving the popularity of NoSQL databases like MongoDB, CouchDB, Cassandra, and HBase.

1. **Storing large volumes of data that often have little to no structure.** A NoSQL database sets no limits on the types of data you can store together, and allows you to add different new types as your needs change. With document-based databases, you can store data in one place without having to define what "types" of data those are in advance.
2. **Making the most of cloud computing and storage.** Cloud-based storage is an excellent cost-saving solution, but requires data to be easily spread across multiple servers to scale up. Using commodity (affordable, smaller) hardware on-site or in the cloud saves you the hassle of additional software, and NoSQL databases like Cassandra are designed to be scaled across multiple data centers out of the box without a lot of headaches.
3. **Rapid development.** If you're developing within two-week Agile sprints, cranking out quick iterations, or needing to make frequent updates to the data structure without a lot of downtime between versions, a relational database will slow you down. NoSQL data doesn't need to be prepped ahead of time.

Now that you've got an overview of SQL vs. NoSQL, who do you need to help you [build and maintain your database systems](#)? Relational and non-relational database management systems can get extremely complicated, and definitely require upkeep—especially when you factor in moving to the cloud. While it's easy to manage a basic single-file database in a program like Microsoft Access, you'll want to hire a capable database architect to handle your relational database management system (RDBMS) or NoSQL database management. Explore [database administrators](#) on Upwork.

Upwork empowers businesses with flexible access to quality talent, on-demand. [Get started today.](#)





Share this article

Upwork is the leading online talent solution. [Learn more](#)



Related Articles

10 Common SQL Programming Mistakes ...

Jennifer Marsh

November 23, 2019 • 17 Min Read

How Much Does it Cost to Hire a SEO...

Brenda Do

October 28, 2019 • 11 Min Read

How Do You Hire a Scientist?

Tyler Keenan

August 21, 2019



Subscribe for Weekly Updates

Get the latest business tips and Upwork news delivered right to your Inbox.

Subscribe

Or follow our [RSS Feed](#).

COMPANY



RESOURCES



BROWSE



FOLLOW US



MOBILE APP





Upwork is the leading online talent solution. [Learn More](#)

