Differences between Sql and NoSql

When it **comes to choosing** a **database** the **biggest decisions** is picking a **relational (SQL)** or **non-relational (NoSQL) data structure**. While **both the databases** are **viable options** still **there** are **certain key differences between the two** that **users must keep in mind** when making a decision.

The main differences:

Type – **SQL databases** are **primarily called** as **Relational Databases (RDBMS)**; whereas **NoSQL database** are **primarily called** as **non-relational** or **distributed database**.

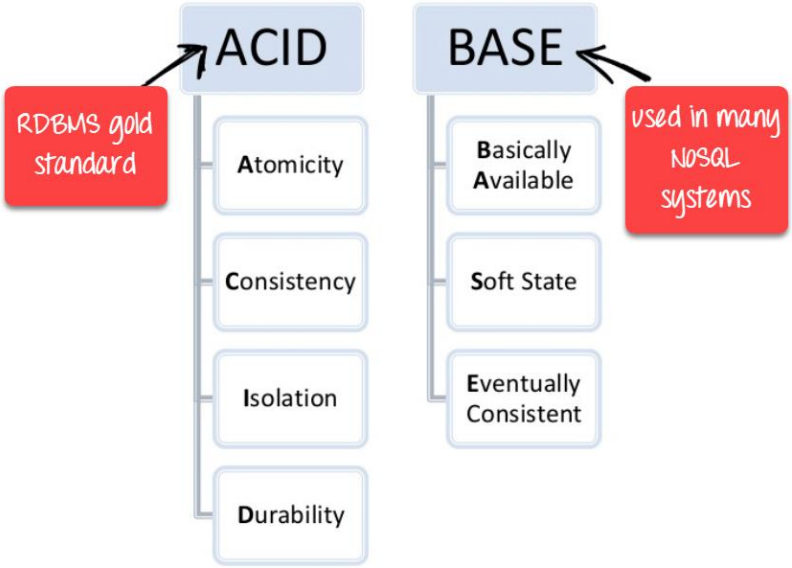
Language – **SQL** databases **defines** and **manipulates** data **based** structured query language **(SQL)**. **SQL** is one of the most **versatile** and **widely-used** options **available** which makes it a **safe choice** especially for **great complex queries**. **But** from other side **it can be restrictive**. **SQL requires** you to use **predefined schemas** to **determine** the **structure of your data** before you work with it. **Also** all of your **data must follow the same structure**. This can **require** significant **up-front preparation** which **means that a change** in the **structure** would be both **difficult** and **disruptive** to your whole system.

A **NoSQL** database has **dynamic schema** for **unstructured data**. **Data** is **stored** in **many ways** which means it **can** be **document-oriented, column-oriented, graph-based** or **organized** as a **KeyValue store**. This **flexibility** means that **documents can be created** without **having**

defined structure first. **Also** each **document** can have **its own unique structure**. The **syntax varies** from **database** to **database**, and you **can add fields as you go**.

The Scalability – In almost all situations **SQL databases** are **vertically scalable**. This means that **you can increase** the **load on a single server** by **increasing** things like **RAM, CPU or SSD**. But on the other hand **NoSQL databases** are **horizontally scalable**. This means that you **handle** more traffic by **sharding**, or **adding more servers** in **your NoSQL database**. It is **similar** to ***adding more floors*** to the ***same building*** versus ***adding more buildings*** to the ***neighborhood***. Thus, **NoSQL** can **ultimately** become **larger** and **more powerful**, making these **databases** the **preferred choice** for **large** or **ever-changing data sets**.

The Structure – **SQL databases** are **table-based** on the other hand **NoSQL databases** are either **key-value pairs, document-based, graph databases** or **wide-column stores**. This makes **relational SQL databases** a **better option** for **applications** that **require multi-row transactions** such as an **accounting system** or for **legacy systems** that were built for a relational structure.



Property followed – **SQL databases** follow **ACID properties** (Atomicity, Consistency, Isolation and Durability) whereas the **NoSQL database** follows the **Brewers CAP theorem** (Consistency, Availability and Partition tolerance).

Support – Great **support** is **available** for all **SQL database** from their vendors. Also, a **lot of independent consultations** are there **who can help** you with **SQL database** for a **very large scale deployments** but for **some NoSQL database** you still **have to rely** on **community support** and **only limited outside experts** are **available** for **setting up** and **deploying** your **large scale NoSQL deployments**.

Some **examples** of **SQL databases** include **PostgreSQL, MySQL, Oracle** and **Microsoft SQL Server. NoSQL** database **examples** include **Redis, RavenDB Cassandra, MongoDB, BigTable, HBase, Neo4j** and **CouchDB**.

Key highlights on Sql vs NoSql:

SQL	NoSQL
RELATIONAL DATABASE MANAGEMENT SYSTEM **(RDBMS)**	**Non-relational** or **distributed database** system.
These **databases** have **fixed** or **static** or **predefined schema**	They have **dynamic schema**
These **databases** are **not suited** for **hierarchical data storage**.	These **databases** are **best suited** for **hierarchical data storage**.
These **databases** are **best suited** for **complex queries**	These **databases** are **not so good** for **complex queries**
Vertically Scalable	**Horizontally** scalable
Follows **ACID** property	Follows **CAP** (consistency, availability, partition tolerance)

Key differences:

- **SQL** pronounced as **"S-Q-L"** or as **"See-Quel"** is **primarily** called **RDBMS** or **Relational Databases** whereas **NoSQL** is a **Non-relational or Distributed Database**.
- Comparing SQL vs NoSQL database, **SQL databases** are **table based databases** whereas **NoSQL databases** can be **document based, key-value pairs, graph databases**.
- **SQL databases** are **vertically scalable** while **NoSQL databases** are **horizontally scalable.**
- **SQL databases** have a **predefined schema** whereas **NoSQL databases** use **dynamic schema for unstructured data**.
- Comparing NoSQL vs SQL performance, **SQL** requires **specialized DB hardware** for **better performance** while **NoSQL** uses **commodity hardware**

Differences between Sql and NoSql:

Parameter	SQL	NOSQL
Definition	SQL databases are primarily called RDBMS or Relational Databases	NoSQL databases are primarily called as Non-relational or distributed database
Design for	Traditional RDBMS uses SQL syntax and queries to analyze and get the data for further insights. They are used for OLAP systems.	NoSQL database system consists of various kind of database technologies. These databases were developed in response to the demands presented for the development of the modern application.
Query Language	Structured query language (SQL)	No declarative query language
Type	SQL databases are table based databases	NoSQL databases can be document based, key-value pairs, graph databases
Schema	SQL databases have a predefined schema	NoSQL databases use dynamic schema for unstructured data.
Ability to scale	SQL databases are vertically scalable	NoSQL databases are horizontally scalable
Examples	Oracle, Postgres, and MS-SQL.	MongoDB, Redis, Neo4j, Cassandra, Hbase.
Best suited for	An ideal choice for the complex query intensive environment.	It is not good fit complex queries.
Hierarchical data storage	SQL databases are not suitable for hierarchical data storage.	More suitable for the hierarchical data store as it supports key-value pair method.
Variations	One type with minor variations.	Many different types which include key-value stores, document databases, and graph databases.
Development Year	It was developed in the 1970s to deal with issues with flat file storage	Developed in the late 2000s to overcome issues and limitations of SQL databases.
Open-source	A mix of open-source like Postgres & MySQL, and commercial like Oracle Database.	Open-source
Consistency	It should be configured for strong consistency.	It depends on DBMS as some offers strong consistency like MongoDB, whereas others offer only offers eventual consistency, like Cassandra.
Best Used for	RDBMS database is the right option for solving ACID problems.	NoSQL is a best used for solving data availability problems

Parameter	SQL	NOSQL
Importance	It should be used when data validity is super important	Use when it's more important to have fast data than correct data
Best option	When you need to support dynamic queries	Use when you need to scale based on changing requirements
Hardware	Specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Network	Highly available network (Infiniband, Fabric Path, etc.)	Commodity network (Ethernet, etc.)
Storage Type	Highly Available Storage (SAN, RAID, etc.)	Commodity drives storage (standard HDDs, JBOD)
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool.
Top Companies Using	Hootsuite, CircleCI, Gauges	Airbnb, Uber, Kickstarter
Average salary	The average salary for any professional SQL Developer is $84,328 per year in the U.S.A.	The average salary for "NoSQL developer" ranges from approximately $72,174 per year
ACID vs. BASE Model	ACID (Atomicity, Consistency, Isolation, and Durability) is a standard for RDBMS	Base (Basically Available, Soft state, Eventually Consistent) is a model of many NoSQL systems

When to use Sql?

- **SQL** is the **easiest language** used to **communicate** with the **RDBMS**
- **Analyzing** behavioral **related** and **customized** sessions
- **Building** custom **dashboards**
- It **allows** you to **store** and **gets data** from the **database quickly**
- **Preferred** when you **want** to **use joins** and **execute complex queries**

When to use NoSql?

- When **ACID** support is **not needed**
- When **Traditional RDBMS model** is **not enough**
- **Data** which **need** a **flexible schema**
- **Constraints** and **validations logic not required** to be **implemented** in **database**
- **Logging data** from **distributed sources**

- It **should be used** to **store temporary data** like **shopping carts, wish list** and **session data**

Pros and Cons of SQL and NoSQL

To **understand** the **clear difference between SQL and NoSQL**, you need to have a comprehensive understanding of the **advantages** and **drawbacks** of **both of them**.

Pros/Advantages of SQL

- **Coding** - No prior knowledge of extensive coding is needed. SQL is simple and easy to learn with declarative commands.
- **Portability** - SQL is highly portable and can be run on Laptops, Mainframes, PC's, tablets or smartphones, etc.
- **Open-source** - SQL is an open-source product that ensures a stronger community for development and assistance in this domain.
- **Interactive Language** - SQL is an interactive language which means it is easy to learn and quite efficient in working with complex queries. Programmers prefer this language as data retrieval in SQL is rapid.
- **Defined standards** - SQL standards are well defined as they were foremostly used by ISO and ANSI.
- **Data Views** - Allows multiple data views
- SQL supports the client-server architecture completely.
- **High Demand** - SQL is in high demand. Many top tier companies are using RDBMS and require administrators with proficiency in SQL skills. Companies like Micorosoft, Hootsuite, Cognizant, and many others are using SQL databases.

Cons/Drawbacks of SQL

- **Complex Structure** - Access is difficult in SQL because of its complex structure.
- **Interfacing** - The process of interfacing can get complicated in SQL as there is not much coding involved in writing SQL queries.

- **Limited** - SQL demand and scaling are limited to particular projects and data types like you cannot use SQL to query unstructured or semi-structured data. It is fixated on a relational schema.

Pros/Advantages of NoSQL

- **Scale-Out Architecture** - NoSQL was designed in 2000 to cover the concerns faced in SQL. It is inevitable to stop the exponential growth of data and to manage that NoSQL provides a scale-out architecture. This technology will allow changing the data plane resources by adding more nodes to the cluster that ensure larger storage capacity and scalability of data.

- **Data Type Storage** - NoSQL flexibility is its biggest advantage. Programmers are not limited to storing only structured data. The freedom from the predefined schema allows NoSQL databases to store and retrieve data easily. From structured data to loosely structured values like maps, strings, binary values etc, can all be stored and retrieved at ease by the data administrators.

- **Developer Friendly** - There is not much hassle in storing and using data for creating applications in NoSQL. Programmers can store data "as it is" i.e., without converting the semi-structured data into a table format for retrieval and manipulation. This enables developers to adapt to structure and technology on their own.

- **Less Management** - NoSQL databases are an enhanced version of a database management system and, thus, does not require a big team of data administrators to perform small tasks or to manage the system. In addition to this, the less complex structure of NoSQL makes it easier to handle and operate.

- **Big Data Applications** - NoSQL databases have a large capacity to store massive volumes of data.

Cons/Drawbacks of NoSQL

- **ACID properties** - Unlike relational databases, NoSQL does not support ACID properties that limit it to provide reliability functions.

- **Query Language** - NoSQL databases do not have any standard query language like SQL. It makes the process of data handling, retrieval, and management more complex and slower.

- **Different Data Models** - NoSQL has different data models for storing and maintaining data like Key-value, document, column family,etc. It hampers NoSQL to facilitate a single database for different purposes instead one has to operate with multiple databases and data models to perform all the niches. This makes the functioning complex, and increases the chances for less data consistency.

- **Huge Databases** - Data quality and Data Duplication are two of the prime issues with NoSQL as it can store up to a massive amount of data without the ACID properties.

SQL vs NoSQL: Which is better?

The biggest and the debatable question to answer.

There is **no denying** the **fact** that both **SQL** and **NoSQL** are some of the **best** of their kind. **SQL** is the **most in-demand programming language** for **RDBMS** and **NoSQL** is the **preferred software** for **storing structured, unstructured** and **semi-structured data**.

And the **answer to which one is better** is – It all **depends** on your **requirements** and the **project** you are working on.

If you are looking for **consistency, reliability**, and a **system to query structured data** you choose **SQL databases**.

However, if you are looking to **work faster** and **independently** for **storing** and **retrieving data** like **graphs, binary numbers**, etc choose **NoSQL databases**.

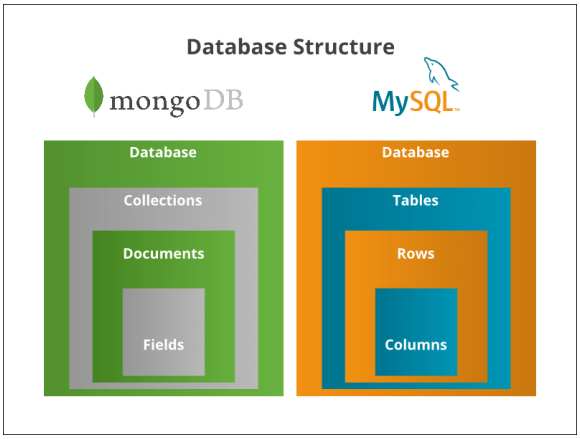
The former focuses on **complex queries**, with **data consistency** and **ACID properties** whereas, the latter is more **object-based** and **suitable for a huge amount** of **different types of data storage**.

Therefore, before **choosing your preferred option** read about the **difference** between **SQL** and **NoSQL thoroughly**.

Differences between MySql and MongoDB?

What is MongoDB?

MongoDB is a **NoSQL** (Not only SQL) **database** that **stores large volumes of data** in the **form of documents**. **MongoDB removes** the **concept of "rows"** of **conventional** and **relational data models** by **introducing "documents."** This offers the **developers** the **flexibility** to **work** with **evolving data models**.



What is MySQL?

MySQL is a free, **open-source**, **relational database management system** that **stores data** in the **form of tables** containing **rows and columns**. It uses **RDBMS** to ensure **referential integrity** between the **rows of a table** and **interprets queries** to **fetch information** from the **database**.

MySQL vs. MongoDB: One-on-one Comparision

Now that you the **objectives** of these **database management systems**, let's look at some of the differences between them.

Feature	MySQL	MongoDB
Data Structure	It stores each individual record as a table cell with rows and columns	It stores unrelated data in JSON like documents
Schema	MySQL requires a schema definition for the tables in the database	MongoDB doesn't require any prior schema

Languages	Supports Structured Query Language (SQL)	Supports JSON Query Language to work with data
Foreign Key	Supports the usage of Foreign keys	Doesn't support the usage of Foreign keys
Replication	Supports master-slave replication and master-master replication	Supports sharding and replication
Scalability	SQL Database can be scaled vertically	MongoDB database can be scaled both vertically and horizontally
Join Operation	Supports Join operation	Doesn't support Join operation
Performance	Optimized for high performance joins across multiple tables	Optimized for write performance
Risks	Prone to SQL injection attack	Since there's no schema, lesser risks involved

Community Support	There are currently (always increasing) about 222k repositories and 7Million commits on GitHubfor support on MySQL	There are currently (always increasing) about 177k repositories and 923k commits on GitHub for support on MongoDB

Which One to Choose?

Applications, like an accounting system that requires multi-row transactions, would be better suited for a relational database. MySQL is an excellent choice if you have structured data and need a traditional relational database.	MongoDB is well-suited for real-time analytics, content management, the Internet of Things, mobile, and other types of applications. It is an ideal choice if you have unstructured and/or structured data with rapid growth potential.