Firebase vs SQLite: What are the differences?

Firebase is a cloud service designed to power real-time, collaborative applications. Simply add the Firebase library to your application to gain access to a shared data structure; any changes you make to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

The main differences between Firebase and SQLite are:

- Firebase belongs to **"Realtime Backend / API"** category of the tech stack, while SQLite can be primarily classified under **"Databases"**.

The main pros for Firebase are:

- Realtime backend
- Fast and responsive
- Easy setup
- Uses JSON
- Free
- Backed by Google

The main pros of SQLite are:

- Is lightweight
- Is portable
- Is simple
- Is sql
- Open source
- Preinstalled on iOS and Android
- Free

The main cons of Firebase:

- Can become expensive (the free version has limitations)
- Scalability is not infinite
- No open source, you depend on google
- Can’t filter queries
- Not flexible enough

The main cons of SQLite:

- Not for multi-process of multi-threaded apps.

While **SQLite** is a good choice for small applications, most of the companies chooses **Firebase** over **SQLite**.

SQL Vs NoSQL: What are the differences?

Structured Query Language, commonly abbreviated as **SQL**, is a domain-specific programming language that is used for storing, manipulating and retrieving data in RDBMS (Relational Database Management System).
It is mainly used for managing structured data where we have a relationship between various entities and variables of the data.

NoSQL (also refers to Not only SQL, non-SQL or non-relational) is a database which gives you a way to manage the data which is in a non-relational form i.e. which is not structured in a tabular manner and does not possess tabular relationships.

SQL Pros:

- It is highly suitable for relational databases.
- Has a predefined schema which is helpful in many cases.
- Normalization can be greatly used here, thus it also helps in removing redundancy and organizing data in a better way.
- Transactions in SQL databases are ACID compliant, thereby guarantees security and stability.
- Follows well-defined standards like ISI and ANSI which are accepted worldwide.
- Code-free.
- Unbeatable speed in retrieving database records with great ease.
- Uses single standardized language i.e SQL across different RDBMS.

SQL Cons:

- The process of interfacing is complex.
- As SQL is an object, it occupies space.
- Handling Big data is very costly as you will have to increase the hardware for scaling.
- When a table is dropped, the view becomes inactive.

NoSQL Pros:

- Capable of handling big data.
- As it is schema-less and table free, it offers a high level of flexibility with data models.
- It is a low-cost database and the open source NoSQL databases provide very affordable solutions to small enterprises.
- Easier and low-cost scalability. You don't need to increase the hardware for scaling. You just need to add more servers to the pool as NoSQL is schema-free and built on distributed systems.
- Detailed database modeling is not required here. Hence it saves time and effort.

NoSQL Cons:

- The benefits of NoSQL come at the cost of relaxing ACID properties. NoSQL offers only eventual consistency.
- Relatively less community support.
- Lacks standardization, unlike SQL, which in turn creates some issues during migration.
- Inter-operability is also a concern in the case of NoSQL databases.

NoSQL databases are gaining large popularity these days due to their capacity to integrate big data, low cost, easy scalability, and open source features. However, it is still a relatively young technology and lacks standardization, unlike **SQL**.