Key Value Database

In this lesson, we will get to know about the Key-Value database and when to choose it for our projects.

We'll cover the following

What Is A Key Value Database?

Key-value databases also are a part of the *NoSQL family*. These databases use a simple *key-value* method to store and quickly fetch the data with minimum latency.

Features Of A Key Value Database

A primary use case of a *Key-value* database is to implement caching in applications due to the minimum latency they ensure.

The *Key* serves as a unique identifier and has a *value* associated with it. The value can be as simple as a block of text & can be as complex as an object graph.

The data in *Key-value* databases can be fetched in *constant time O(1)*, there is no query language required to fetch the data. It's just a simple no-brainer fetch operation. This ensures minimum latency.

Popular Key Value Databases

Some of the popular *key-value* data stores used in the industry are *Redis*, *Hazelcast*, *Riak*, *Voldemort* & *Memcache*.

When Do I Pick A Key Value Database?

If you have a use case where you need to fetch data real fast with minimum fuss & backend processing then you should pick a *key-value* data store.

Key-value stores are pretty efficient in pulling off scenarios where super-fast data fetch is the order of the day.

Typical use cases of a *key value* database are the following:

- Caching
- Persisting user state
- Persisting user sessions
- Managing real-time data
- Implementing queues
- Creating leaderboards in online games & web apps
- Implementing a pub-sub system

Real Life Implementations

Some of the real-life implementations of the tech are -

- Inovonics uses Redis to drive real-time analytics on millions of sensor data
- Microsoft uses Redis to handle the traffic spike on its platforms
- Google Cloud uses Memcache to implement caching on their cloud platform