Features Of NoSQL Databases

In this lesson, we will discuss the features of NoSQL databases.

WE'LL COVER THE FOLLOWING

In the introduction we learned that the *NoSQL* databases are built to run on clusters in a distributed environment, powering Web 2.0 websites.

Now, let's go over some features of *NoSQL* databases.

Pros Of NoSQL Databases

Besides the design part, *NoSQL* databases are also developer-friendly. What do I mean by that?

Gentle Learning Curve

First, the learning curve is less than that of relational databases. When working with relational databases a big chunk of our time goes into learning how to design well-normalized tables, setting up relationships, trying to minimize joins and stuff.

Schemaless

One needs to be pretty focused when designing the schema of a relational database to avoid running into any issues in the future.

Think of relational databases as a strict headmaster. Everything has to be in place, neat and tidy, things need to be consistent. But *NoSQL* databases are a bit of chilled out & relaxed.

There are no strict enforced schemas, work with the data as you want. You can always change stuff, spread things around. Entities have no relationships. Thus, things are flexible & you can do stuff your way.

Wonderful Right?

Not always!! This flexibility is good and bad at the same time. Being so flexible, developer-friendly, having no joins and relationships etc. makes it good.

Cons Of NoSQL Databases

Inconsistency

But it introduces a risk of entities being inconsistent at the same time. Since an entity is spread throughout the database one has to update the new values of the entity at all places.

Failing to do so, makes the entity inconsistent. This is not a problem with relational databases since they keep the data normalized. An entity resides at one place only.

No Support For ACID Transactions

Also, *NoSQL* distributed databases don't provide *ACID transactions*. A few that claim to do that, don't support them globally. They are just limited to a certain entity hierarchy or a small region where they can lock down nodes to update them.

> **Note:** *Transactions in distributed systems come with terms and conditions applied.*

Conclusion

My first experience with a *NoSQL* datastore was with the *Google Cloud Datastore*.

An upside I felt was that we don't have to be a pro in database design to write an application. Things were comparatively simpler, as there was no stress of managing joins, relationships, n+1 query issues etc.

Just fetch the data with its Key. You can also call it the id of the entity. This is a *constant O(1)* operation, which makes *NoSQL* Dbs really fast.

I have designed a lot of *MySQL* DB schemas in the past with complex relationships. And I would say working with a *NoSQL* database is a lot easier than working with relationships.

It's alright if we need to make a few extra calls to the backend to fetch data in separate calls that doesn't make much of a difference. We can always cache the frequently accessed data to overcome that.

Popular NoSQL Databases

Some of the popular NoSQL databases used in the industry are *MongoDB*, *Redis*, *Neo4J*, *Cassandra*.

So, I guess, by now we have a pretty good idea of what NoSQL databases are. Let's have a look at some of the use cases which fit best with them.