

Web Application & Software Architecture 101

0% completed

Database

- Introduction & Types of Data
- Relational Database
- When Should You Pick A Relational Database?**
- NoSQL Databases - Introduction
- Features Of NoSQL Databases
- When To Pick A NoSQL Database?
- Is NoSQL More Performant than SQL?
- Database Quiz - Part 1
- Polyglot Persistence
- Multi-Model Databases
- Eventual Consistency
- Strong Consistency
- CAP Theorem
- Database Quiz - Part 2
- Types of Databases
- Document Oriented Database
- Graph Database
- Key Value Database
- Time Series Database
- Wide-Column Database
- Database Quiz - Part 3

Caching

- Introduction
- How Do I figure If I Need A Cache In My Application?
- Reducing the Application Deployment Costs Via Caching
- Caching Strategies
- Scalable Caching
- Database Quiz - Part 1
- Polyglot Persistence
- Multi-Model Databases
- Eventual Consistency
- Strong Consistency
- CAP Theorem
- Database Quiz - Part 2
- Types of Databases
- Document Oriented Database
- Graph Database
- Key Value Database
- Time Series Database
- Wide-Column Database
- Database Quiz - Part 3

KB



Explore



Tracks

When Should You Pick A Relational Database?

In this lesson, we will discuss when to choose a relational database for our project.

We'll cover the following

- Transactions & Data Consistency
- Large Community
- Storing Relationships
- Popular Relational Databases

If you are writing a stock trading, banking or a Finance-based app or you need to store a lot of relationships, for instance, when writing a social network like *Facebook*. Then you should pick a relational database. Here is why –

Transactions & Data Consistency

If you are writing a software which has anything to do with money or numbers, that makes *transactions*, *ACID*, *data consistency* super important to you.

Relational DBs shine when it comes to transactions & data consistency. They comply with the ACID rule, have been around for ages & are battle-tested.

Large Community

Also, they have a larger community. Seasoned engineers on the tech are easily available, you don't have to go too far looking for them.

Storing Relationships

If your data has a lot of relationships like which friends of yours live in a particular city? Which of your friend already ate at the restaurant you plan to visit today? etc. There is nothing better than a relational database for storing this kind of data.

Relational databases are built to store relationships. They have been tried & tested & are used by big guns in the industry like [Facebook as the main user-facing database](#).

Popular Relational Databases

Some of the popular relational databases used in the industry are *MySQL* - it's an open-source relationship database written in C, C++ been around since 1995.

Others are *Microsoft SQL Server*, a proprietary RDBMS written by Microsoft in C, C++. *PostgreSQL* an relationships, for instance, when writing a social network like *Facebook*. Then you should pick a relational database. Here is why –

Transactions & Data Consistency

If you are writing a software which has anything to do with money or numbers, that makes *transactions*, *ACID*, *data consistency* super important to you.

Relational DBs shine when it comes to transactions & data consistency. They comply with the ACID rule, have been around for ages & are battle-tested.

Large Community

Also, they have a larger community. Seasoned engineers on the tech are easily available, you don't have to go too far looking for them.

Storing Relationships