(/learn)

When should you pick a relational database?

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

We'll cover the following

- Transactions and data consistency
- Large community
- Storing relationships
- Popular relational databases

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

Transactions and data consistency#

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

Relational DBs shine when it comes to transactions and data consistency.

They comply with the ACID rule, have been around for ages, and are battle-tested.

Large community#

Additionally, they have a larger community. Seasoned engineers on the tech are easily available, so 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 friends 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 and tested and are used by big guns in the industry, like Facebook, as the main user-facing database. (https://www.8bitmen.com/what-database-does-facebook-use-a-1000-feet-deep-dive/)

Popular relational databases#

Some of the popular relational databases used in the industry are *MySQL*, which is an open-source relationship database written in *C* and *C++*, which has been around since 1995.

Others are *Microsoft SQL Server*, a proprietary RDBMS written by Microsoft in C and C++. *PostgreSQL* is an open-source RDBMS written in C. Additionally, there are *MariaDB*, *Amazon Aurora*, *Google Cloud SQL* etc.

Well, that’s all on the relational databases. Moving on to non-relational databases.

← Back

Relational Databases

Next →

NoSQL Databases - Introduction

✓ Mark as Completed

Report an Issue