Polyglot Persistence

In this lesson, we will break down what is meant by polyglot persistence.

We'll cover the following

What Is polyglot persistence?

> *Polyglot persistence* means using several different persistence technologies to fulfil different persistence requirements in an application.

We will discuss this concept with the help of an example.

Real-world use case#

Let’s say we are designing a social network like Facebook.

Relational database#

To store relationships like persisting friends of a user, friends of friends, what rock band user likes, what food preferences users have in common with friends, etc., we would pick a relational database like *MySQL*.

Key Value Store#

For low latency access of all the frequently accessed data, we will implement a cache using a *Key-value* store like *Redis* or *Memcached*.

We can use the same *Key-value* data store to store user *sessions*.

Now, our app is already a big hit. It has gotten pretty popular, and we have millions of active users.

Wide column database#

To understand user behavior, we need to set up an analytics system to run analytics on the data generated by the users. We can do this using a *wide-column* database like *Cassandra* or *HBase*.

ACID transactions and strong consistency#

The popularity of our application just doesn't seem to stop, it's soaring. Now businesses want to run ads on our portal. For this, we need to set up a payment system.

Again, we would pick a *relational database* to implement *ACID transactions* and ensure *strong consistency*.

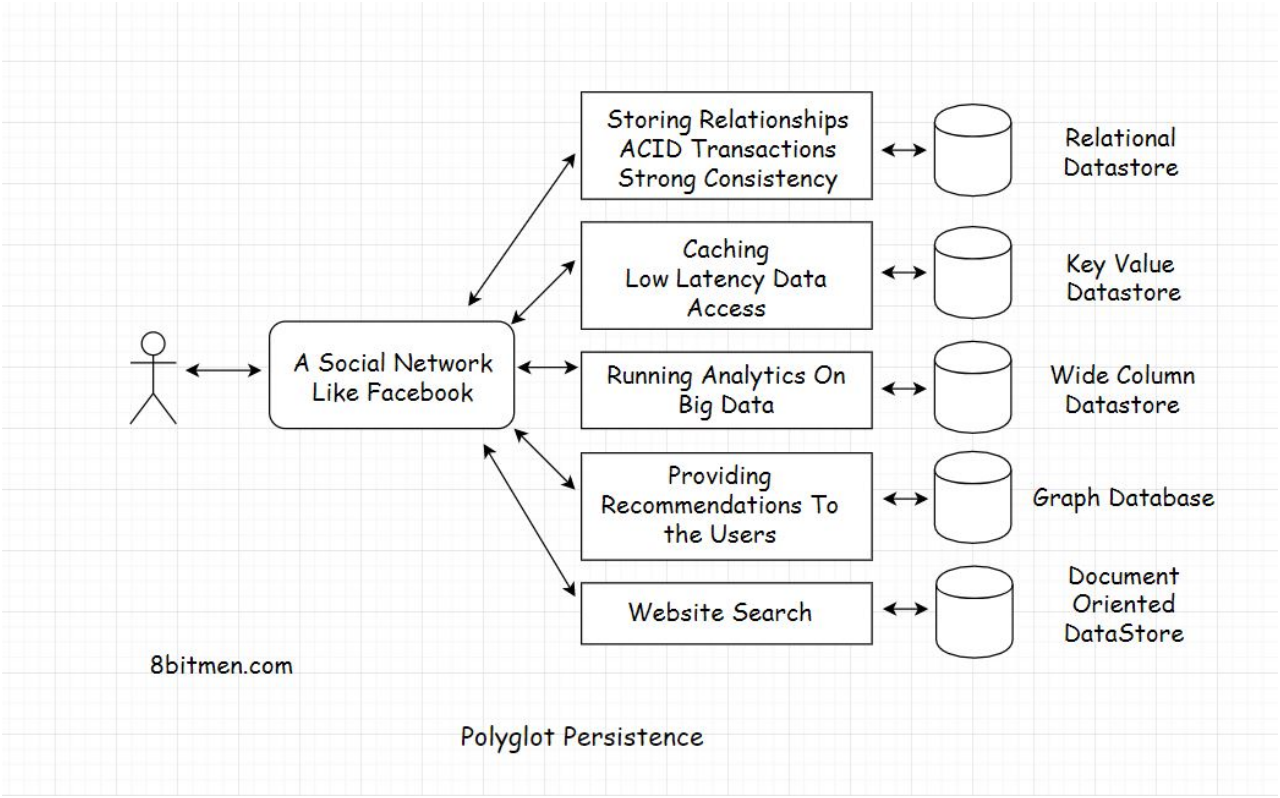
Graph database#

Now to enhance our application's user experience, we have to start recommending content to the users to keep them engaged. A *graph database* would fit best to implement a recommendation system.

Alright, by now, our application has multiple features, and everyone loves it. How cool would it be if a user could run a search for other users, business pages, and what not on our portal and connect with them?

Document Oriented Store#

To implement this, we can use an open-source *document-oriented* datastore like *Elasticsearch*. The product is pretty popular in the industry for implementing a scalable search feature on websites. We can persist all the search-related data in the elastic store.



Downside of this approach#

So, this is how we use multiple databases to fulfil different persistence requirements. However, one downside of this approach is the increased complexity required to make all these different technologies work together.

A lot of effort goes into building, managing and monitoring polyglot persistence systems. What if there was something simpler? That would save us the pain of putting together everything ourselves. Well, there is.

What?

You will find out what this is in the next lesson.

← Back

Next →

Database Quiz - Part 1

Multi-Model Databases

Mark as Completed

Report an Issue