Polyglot Persistence

In this lesson, we will understand 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 understand this concept with the help of an example.

Real World Use Case

Let's say we are writing a social network like Facebook.

Relational Database

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

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 *Memcache*.

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

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

Wide Column Database

To understand user behaviour, 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 & 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 payments system.

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

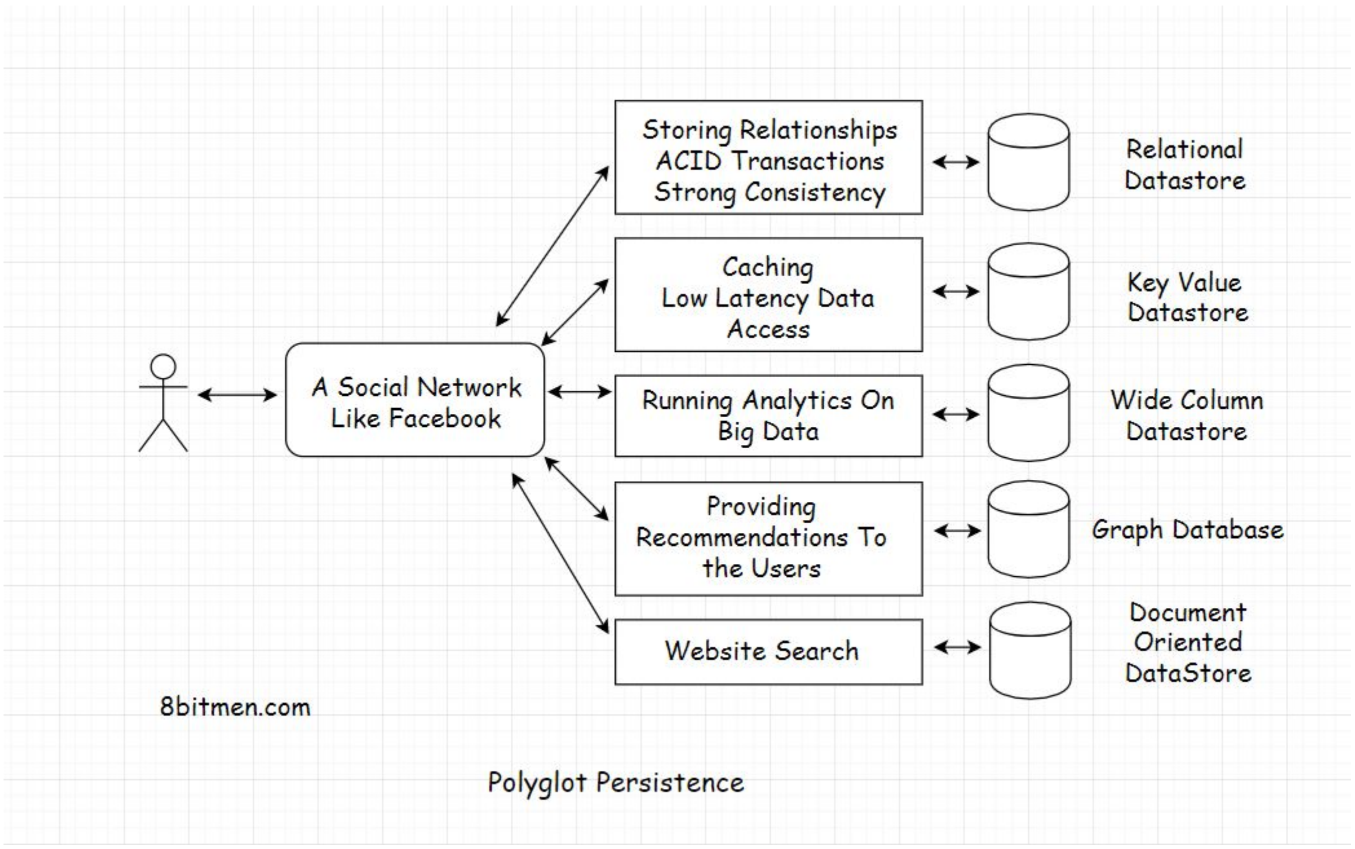
Graph Database

Now to enhance the user experience of our application 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, & everyone loves it. How cool it would be if a user can run a search for other users, business pages and stuff on our portal & could 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. Though, one downside of this approach is increased complexity in making 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?

Let's find out in the next lesson.