Literature survey of Databases

For our project CorpConnect, we need two main databases:

1. Users database: This database will store the information about the users like Name, Job ID etc.

2. Posts database: This database will store the actual documents that a user uploads.

We have a variety of different approaches for types of databases in the market.

Relational Databases:

These are the types of databases that store information in tables and columns. It is the earliest form of database model. They are defined by ACID properties of atomicity, consistency, isolation, and durability. These types of databases are best suited to working with structured data only. They allow us to create relationships between different types of data and perform various types of analysis on the data easily. Some examples of relational databases are Microsoft SQL Server, MySQL, DB2, PostgreSQL.

Non-Relational Databases:

Non-relational databases (or NoSQL) store data in different formats like key-value pairs, documents, graphs etc. It is suitable for use when we need to handle large amounts of unstructured and semi-structured data. They provide flexible schema designs that allow efficient storage and retrieval. Traditional relational databases are designed to work on single servers only, leading to performance issues when handling large amounts of data.

For our project needs we can use a combination of SQL and noSQL databases.

We can use a relational database to store the data in the Users database. Since the users data will only contain simple values like user name, job id etc., this data can be easily contained in a structured way in tables and columns.

For the Posts database, this database will need to store all the documents that a user posts. There could be huge number of documents and we also need the ability to search in the documents efficiently. This use-case may not be handled by relational databases efficiently. To support this large amount of data and easy searchability, a noSQL database can be used to satisfy this requirement.

Let’s explore the various NoSQL databases available:

There are various NoSQL databases like Key-value pair, Document-oriented, Column-oriented, Graph-based, Time series etc. For our requirement of storing documents, we require a Document database. Document databases have a flexible schema, meaning that not all documents in a collection need to have the same fields.

Below are some of the options:

MongoDB:

MongoDB is an open-source document-oriented database that is well-suited for storing high-volume data. It allows you to store documents (in BSON format) with flexible schemas. MongoDB allows flexibility, scalability, and powerful querying capabilities.

Key features:

Scalability: MongoDB can handle large datasets and provides horizontal scaling through sharding.
Indexing: Supports various indexing options for efficient searching.
Rich queries: Allows complex queries and aggregation pipelines.

Amazon DynamoDB:

Suitability: DynamoDB is a fully managed NoSQL database service provided by AWS, offering seamless scalability, high availability, and low latency. It's suitable for use cases where you need predictable performance and automatic scaling without managing infrastructure.

Advantages: DynamoDB offers seamless scalability, automatic scaling, and built-in features like multi-region replication and backup. It provides predictable performance with configurable read and write throughput.

Considerations: DynamoDB has more limited querying capabilities compared to MongoDB. While it's well-suited for key-value access patterns and simple queries, complex queries may require additional processing or data modeling.

Elasticsearch:

Although Elasticsearch is primarily known as a search engine, it can also serve as a document store. Elasticsearch is designed for full-text search, analytics, and real-time data. We can use Elasticsearch if we want to prioritize search functionality and real-time indexing.

Full-text search: Provides powerful text search capabilities.
Distributed architecture: Scales horizontally across nodes.
Real-time indexing: Suitable for near real-time data.
Aggregations: Supports complex aggregations.

Based on the above, MongoDB is the suitable choice to store the Posts database.