Session 10: NOSQL

Topic: Introduction to NoSQL

NoSQL is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data. NoSQL databases store data in documents rather than relational tables. NoSQL databases use a variety of data models for accessing and managing data.

The acronym NoSQL was first used in 1998 by Carlo Strozzi while naming his lightweight, open-source "relational" database that did not use SQL. The name came up again in 2009 when Eric Evans and Johan Oskarsson used it to describe non-relational databases.

NoSQL databases are generally classified into four main categories:

1. **Document databases:** These databases store data as semi-structured documents, such as JSON or XML, and can be queried using document-oriented query languages. **MongoDB** is a popular option for developers looking to work with a document database.
2. **Key-value stores:** These databases store data as key-value pairs, and are optimized for simple and fast read/write operations.

```
      Key                                   Value
customer-123 { "address": "…", name: "…", "preferences": {…} }
customer-456 { "address": "…", name: "…", "preferences": {…} }
```

3. **Column-family stores:** These databases store data as column families, which are sets of columns that are treated as a single entity. They are optimized for fast and efficient querying of large amounts of data.

Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

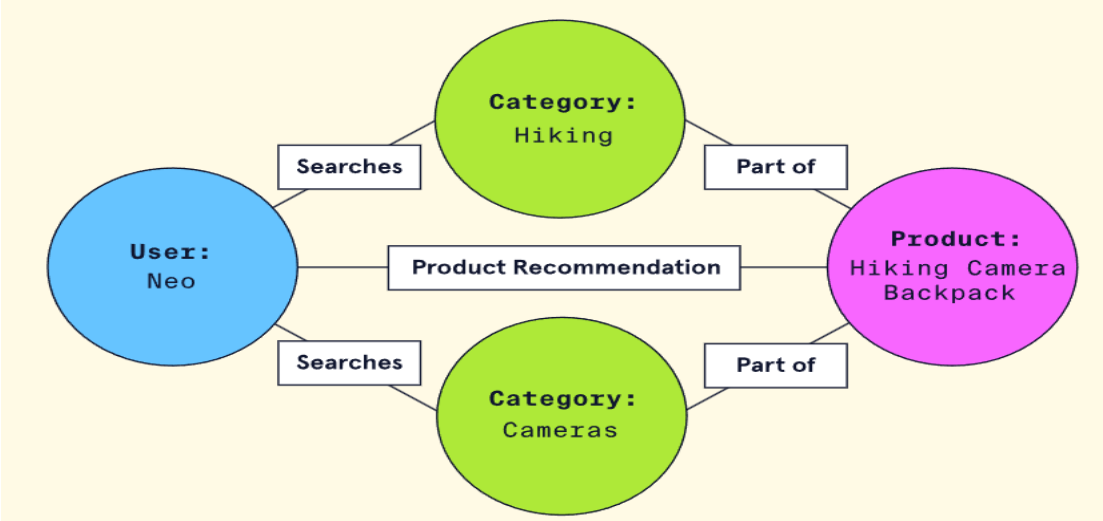
Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

4. **Graph databases:** These databases store data as nodes and edges, and are designed to handle complex relationships between data.

   NoSQL databases are often used in applications where there is a high volume of data that needs to be processed and analysed in real-time, such as social media analytics, e-commerce, and gaming. They can also be used for other applications, such as content management systems, document management, and customer relationship management.



Key Features of NoSQL:

A. **Dynamic schema:** NoSQL databases do not have a fixed schema and can accommodate changing data structures without the need for migrations or schema alterations.

B. **Horizontal scalability:** NoSQL databases are designed to scale out by adding more nodes to a database cluster, making them well-suited for handling large amounts of data and high levels of traffic.

C. **Document-based:** Some NoSQL databases, such as MongoDB, use a document-based data model, where data is stored in a **scales** semi-structured format, such as JSON or BSON.

D. **Key-value-based:** Other NoSQL databases, such as Redis, use a key-value data model, where data is stored as a collection of key-value pairs.

E. **Column-based:** Some NoSQL databases, such as Cassandra, use a column-based data model, where data is organized into columns instead of rows.

F. **Distributed and high availability:** NoSQL databases are often designed to be highly available and to automatically handle node failures and data replication across multiple nodes in a database cluster.

G. **Flexibility:** NoSQL databases allow developers to store and retrieve data in a flexible and dynamic manner, with support for multiple data types and changing data structures.

H. **Performance:** NoSQL databases are optimized for high performance and can handle a high volume of reads and writes, making them suitable for big data and real-time applications.

Advantages of NoSQL: There are many advantages of working with NoSQL databases:

1. **High scalability:** NoSQL databases use sharding for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharding.

   Examples of horizontal scaling databases are MongoDB, Cassandra, etc. NoSQL can handle a huge amount of data because of scalability, as the data grows NoSQL scales The auto itself to handle that data in an efficient manner.

2. **Flexibility:** NoSQL databases are designed to handle unstructured or semi-structured data, which means that they can accommodate dynamic changes to the data model. This makes NoSQL databases a good fit for applications that need to handle changing data requirements.

3. **High availability:** The auto**,** replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.

4. **Scalability:** NoSQL databases are highly scalable, which means that they can handle large amounts of data and traffic with ease. This makes them a good fit for applications that need to handle large amounts of data or traffic

5. **Performance:** NoSQL databases are designed to handle large amounts of data and traffic, which means that they can offer improved performance compared to traditional relational databases.

6. **Cost-effectiveness:** NoSQL databases are often more cost-effective than traditional relational databases, as they are typically less complex and do not require expensive hardware or software.

7. **Agility:** Ideal for agile development.

Disadvantages of NoSQL: NoSQL has the following disadvantages.

1. **Lack of standardization:** There are many different types of NoSQL databases, each with its own unique strengths and weaknesses. This lack of standardization can make it difficult to choose the right database for a specific application
2. **Lack of ACID compliance:** NoSQL databases are not fully ACID-compliant, which means that they do not guarantee the consistency, integrity, and durability of data. This can be a drawback for applications that require strong data consistency guarantees.
3. **Narrow focus:** NoSQL databases have a very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.
4. **Open-source:** NoSQL is a **database** open-source database. There is no reliable standard for NoSQL yet. In other words, two database systems are likely to be unequal.
5. **Lack of support for complex queries:** NoSQL databases are not designed to handle complex queries, which means that they are not a good fit for applications that require complex data analysis or reporting.
6. **Lack of maturity:** NoSQL databases are relatively new and lack the maturity of traditional relational databases. This can make them less reliable and less secure than traditional databases.
7. **Management challenge:** The purpose of big data tools is to make the management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than in a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.
8. **GUI is not available:** GUI mode tools to access the database are not flexibly available in the market.
9. **Backup:** Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.
10. **Large document size:** Some database systems like MongoDB and CouchDB store data in JSON format. This means that documents are quite large (BigData, network

bandwidth, speed), and having descriptive key names actually hurts since they increase the document size.

Topic2: Difference between DBMS, RDBMS and a NoSQL database

Parameter	DBMS	RDBMS	NoSQL
Storage	Data stored as a file system.	RDBMS applications store data in the form of table structured manner.	NoSQL is a non-relational database system. It stores data in the form of unstructured. manner.
Number of users	It supports a single user.	RDBMS supports multiple users.	It also supports multiple users.
Database structures	In DBMS, data stores either in either a navigational or hierarchical form.	RDBMS uses tabular structures to store data. In table headers are the column names and the rows contains corresponding values.	NoSQL uses to store data in structured, semi-structured and unstructured forms.
ACID	In regular DBMS, the data not be stored following the ACID. It creates an inconsistencies database.	RDBMS are harder to construct and obey ACID (Atomicity, Consistency, Isolation, Durability). It helps to create consistency database.	NoSQL may support ACID to store data.
Normalization	The database does not support normalization.	It supports the normalization and joining of tables.	It does have table form, so it does not support normalization.
open-source	It's a mix of an open-source and commercial database.	Open-source application	Open-source program

Integrity constraints	DBMS database does not support the integrity constants.	The relational database supports the integrity constraints at the schema level. Data values beyond a defined range cannot be stored in the particular RDBMS column.	NoSQL database supports integrity constraints.
Development year	In the 1960s DBMS is introduced to store data.	It was developed in the 1970s to deal with the issues of flat file storage.	It developed in the late 2000s to overcome the issues and limitations of the SQL database.
Distributed database	It does not support a distributed database.	It supports a distributed database.	It also supports a distributed database.
Ideally suited for	This system deals with a small quantity of data.	This database system deals with a large quantity of data.	NoSQL database mainly designed for Big data and real-time web data.
Client-server	DBMS system does not support client-server architecture.	RDBMS program support client-server architecture.	NoSQL storage system supports multi-server. It also supports client-server architecture.
Data relationship	No relationship between the data value	Data related to each other with the help of foreign keys	Data can be stored in a single document file.
Hardware and software	Low software and hardware	High software and specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Data fetching	Data fetching is slower for complex data.	Data fetching is rapid because of its relational approach and database.	Data fetching is easy and flexible.
Examples	XML, Windows Registry, file system, etc.	MYSQL, Oracle, SQL Server, etc.	Apache HBase, IBM Domino,Oracle NoSQL Database,etc.

Understanding the Storage Architecture of NOSQL Database

NoSQL databases provide four different data model storage, each one of which has its own advantages and limitations (Key-Value, column data model, document data model, and graph data model. Non-relational databases are powerful storage systems that emerged in the early 2000s. They're designed to manage large volumes of unstructured data, return real-time web app analytics, and process big data across the internet of things (IoT).

The primary ways data can be stored in NoSQL include:

- **Key-Value Store**
- **Document Store**
- **Column Store**
- **Graph Store**

Topic: Working with Column-Oriented Databases

The Columnar Data Model of NoSQL is important. NoSQL databases are different from SQL databases. NoSQL databases are a flexible schema model which is designed to scale horizontally across many servers and is used in large volumes of data.

Columnar Data Model of NoSQL:

The relational database stores data in rows and also reads the data row by row, column store is organized as a set of columns.

Examples of Columnar Data Model: Cassandra and **Apache Hadoop Hbase.**

column – Oriented Table:

S.No.	Name	ID
01.	Tanmay	2
02.	Abhishek	5
03.	Samriddha	7
04.	Aditi	8

S.No.	Branch	ID
01.	Computer	2
02.	Electronics	5
03.	IT	7
04.	E & TC	8

Columnar Data Model uses the concept of key space, which is like a schema in relational models.

Advantages of Columnar Data Model:

- **Well structured:** Since these data models are good at compression so these are very structured or well organized in terms of storage.

- **Flexibility:** A large amount of flexibility as it is not necessary for the columns to look like each other, which means one can add new and different columns without disrupting the whole database
- **Aggregation queries are fast:** The most important thing is aggregation queries are quite fast because a majority of the information is stored in a column. An example would be Adding up the total number of students enrolled in one year.
- **Scalability:** It can be spread across large clusters of machines, even numbering in thousands.
- **Load Times:** Since one can easily load a row table in a few seconds so load times are nearly excellent.

Disadvantages of Columnar Data Model:

- **Designing indexing Schema:** To design an effective and working schema is too difficult and very time-consuming.
- **Suboptimal data loading:** incremental data loading is suboptimal and must be avoided, but this might not be an issue for some users.
- **Security vulnerabilities:** If security is one of the priorities, then it must be known that the Columnar data model lacks inbuilt security features in this case, one must look into relational databases.
- **Online Transaction Processing (OLTP):** Online Transaction Processing (OLTP) applications are also not compatible with columnar data models because of the way data is stored.

Applications of Columnar Data Model:

- Columnar Data Model is very much used in various Blogging Platforms.
- It is used in Content management systems like WordPress, Joomla, etc.
- It is used in Systems that maintain counters.
- It is used in Systems that require heavy write requests.
- It is used in Services that have expiring usage.

Topic: Document Store Internals

A document database is a type of NoSQL database that can be used to store and query data as JSON-like documents. JavaScript Object Notation (JSON) is an open data interchange

format that is both human and machine-readable. Developers can use JSON documents in their code and save them directly into the document database. The flexible, semi-structured, and hierarchical nature of documents and document databases allows them to evolve with applications' needs.

How do document databases work

Document databases store data as key-value pairs in JSON format. You can read and write JSON documents to the databases programmatically.

JSON document's structure: JSON represents data in three ways:

A. Key value

Key-value pairs are recorded within curly braces. The key is a string, and the value can be any data type like integer, decimal, or boolean.

For example, a simple key-value is {"year": 2013}.

B. Array

An array is an ordered collection of values defined within left ([) and right (]) brackets. Items in the array are comma separated. For example, {"fruit": ["apple","mango"]}.

C. Objects

An object is a collection of key-value pairs. Essentially, JSON documents allow developers to embed objects and create nested pairs. For example, {"address": {"country": "USA","state": "Texas"}}.

Document database operations

You can create, read, update, and delete entire documents stored in the database. Document databases provide a query language or API that allows developers to run the following operations:

1. **Create: you** can create documents in the database. Each document has a unique identifier that serves as a key.
2. **Read:** You can use the API or query language to read document data. You can run queries using field values or keys. You can also add indexes to the database to increase read performance.
3. **Update: You** can update existing documents flexibly. You can rewrite the entire document or update individual values.

What is Write-Ahead Logging (WAL)?

Write-ahead logging is a protocol that requires changes to data to be written to a log before the corresponding data is updated in the main storage.

The idea is that before any modifications are made to the actual database or file system, a record of these changes is written to a log file.

Once the log entry is successfully written to the disk, the corresponding changes can be applied to the main storage. It is an append-only mechanism and ensures data integrity.

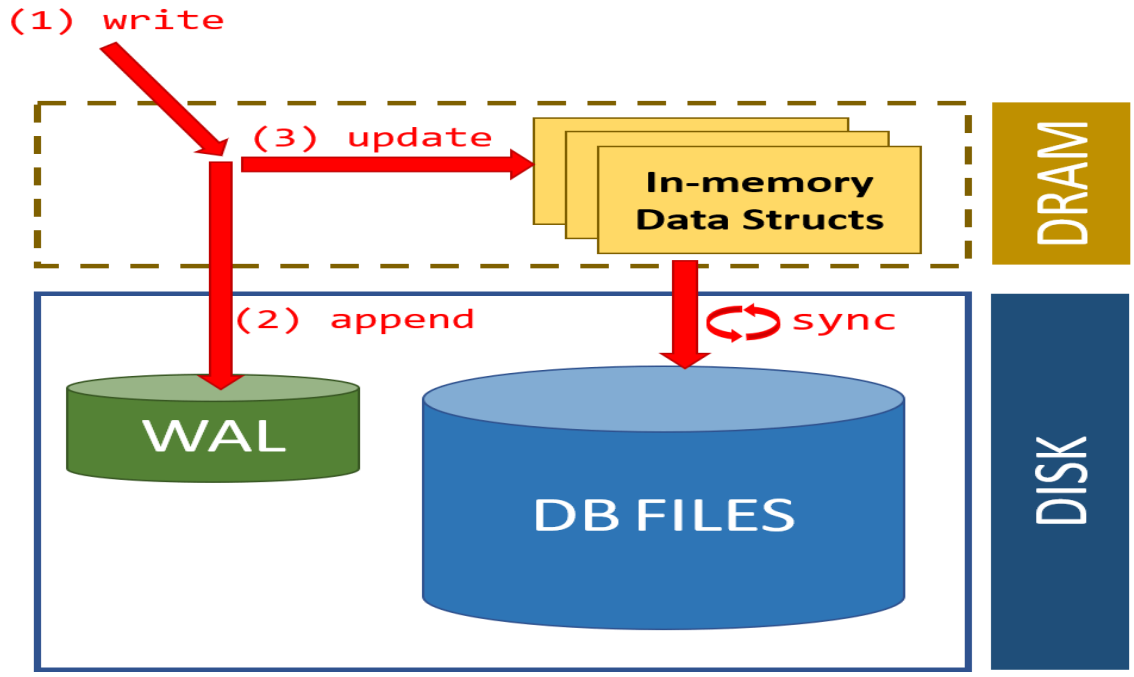


Fig- Use of WAL

Why is it used for?

1. **Durability**: By writing the changes to the log first, the system ensures that even if a crash occurs or power is lost, the modifications are not lost. During recovery, the system can use the log to bring the data back to a consistent state by replaying the logged changes.
2. **Atomicity**: Write-ahead logging is often used in transactional systems to maintain atomicity. Atomicity ensures that a transaction is treated as a single, indivisible unit of work. If any part of the transaction fails, the entire transaction is rolled back, and the database remains in a consistent state.
3. **Performance**: While writing to the disk is generally slower compared to writing to memory, write-ahead logging can improve performance by batching multiple updates into a single write operation. This reduces the number of disk writes, which can be a significant

performance bottleneck. Also, let us say there is one byte in a page that needs to be changed. There is no byte addressability on disk; it is possible only on RAM, so data needs to be written in chunks.

4. **Crash Recovery**: In the case of system crashes, the system can use the write-ahead log to recover the database or file system to a consistent state. By replaying the logged changes, the system can reconstruct the lost data and bring the system back online without data corruption.

How write-ahead logging is typically used:

1. **Start of Transaction**: When a transaction begins, the system creates a new log entry to mark the start of the transaction.
2. **Modifications**: As the transaction progresses and data is modified or updated, the changes are recorded in the log file.
3. **Commit**: When the transaction is ready to be committed, a special log entry called "commit record" is written to the log, indicating that all changes associated with the transaction are now considered durable.
4. **Apply Changes**: After the commit record is successfully written to the log, the changes are applied to the main database or file system. This ensures that data remains consistent, and the log acts as a safety net in case of a crash before the changes are permanently stored.
5. **Recovery**: In the event of a system crash or failure, the system can use the write-ahead log to recover lost data. During recovery, the log is replayed, and the changes are reapplied to the main storage, bringing the system back to a consistent state.

Examples of Document Data Models:

- Amazon DocumentDB
- MongoDB
- Cosmos DB
- ArangoDB
- Couchbase Server
- CouchDB

Advantages:

- **Schema-less:** These are very good in retaining existing data at massive volumes because there are absolutely no restrictions in the format and the structure of data storage.
- **Faster creation of document and maintenance:** It is very simple to create a document and apart from this maintenance requires is almost nothing.
- **Open formats:** It has a very simple build process that uses XML, JSON, and its other forms.
- **Built-in versioning:** It has built-in versioning which means as the documents grow in size there might be a chance they can grow in complexity.

Disadvantages:

- **Weak Atomicity:** It lacks in supporting multi-document ACID transactions. A change in the document data model involving two collections will require us to run two separate queries i.e. one for each collection. This is where it breaks atomicity requirements.
- **Consistency Check Limitations:** One can search the collections and documents that are not connected to an author collection but doing this might create a problem in the performance of database performance.
- **Security:** Nowadays many web applications lack security which in turn results in the leakage of sensitive data.

Applications of Document Data Model:

- **Content Management:** These data models are very much used in creating various video streaming platforms, blogs, and similar services Because each is stored as a single document and the database here is much easier to maintain as the service evolves over time.
- **Book Database:** These are very much useful in making book databases because as we know this data model lets us nest.
- **Catalog:** When it comes to storing and reading catalogue files these data models are very much used because it has a fast-reading ability if incase Catalogs have thousands of attributes stored.
- **Analytics Platform:** These data models are very much used in the Analytics Platform.