Name: Kartik Vedi Student ID: 300374518  
CSIS 4260 Section: 01 Seminar: 4

## Topic: NoSQL Databases (Excluding MongoDB)

# Executive Summary

We examined the dynamic world of NoSQL databases in this lecture, with an emphasis on how they manage unstructured data and scale horizontally. The main conclusions are as follows:

1. **Overview of NoSQL Databases:**

o NoSQL databases, such as Cassandra and DynamoDB, provide more flexibility than conventional relational databases.

o They work best in situations where data needs to scale seamlessly and comes in a variety of forms.

2**. NoSQL Trends:**

o Graph databases: Neo4j and related programmes are excellent at simulating intricate relationships. Contemplate recommendation engines and social networks.

o Time-Series Databases: For real-time analytics and the Internet of Things, InfluxDB effectively handles time-stamped data.

o Multi-Model Databases: These databases offer versatility by combining features from several NoSQL models.

Organisations can address the issues brought on by ever expanding data quantities by carefully implementing NoSQL databases. NoSQL gives us the tools to traverse the data-driven future, whether it be for collecting sensor data, optimising recommendation engines, or guaranteeing real-time insights.

Recall that the NoSQL world is always changing, thus being up to date is crucial. Continue your research, testing, and trend-following.

# Main Idea

## **Introduction**

NoSQL databases capacity to handle a wide range of data types and their scalability, flexibility, and adaptability have completely changed the data management industry. This section will examine the fundamental concept of NoSQL databases and go over its practical uses.

### **Example Scenario: E-Commerce Recommendation Engine**

Think of an online store that seeks to offer its customers customised product recommendations. Relational databases have always had difficulty keeping up with the dynamic nature of real-time updates, product interactions, and user preferences. This is the sweet spot for NoSQL databases.

## The Algorithm: Building a Recommendation Engine

1. **Data Modeling**:

o Because graph databases, like Neo4j, can depict intricate relationships, that is why we have chosen them.

o Nodes stand for people, things, and interactions (likes, purchases, and views).

o Edges identify different forms of interactions and link users to products.

1. **Graph Traversal**:

o We search the graph for a user's associated products after they log in.

o We take into account both direct interactions, like purchases, and indirect relationships, like preferences of similar users.

1. **Recommendation Generation**:

o We find similar people based on the user's graph neighbourhood.

o We suggest products that the present user hasn't used but these comparable users have.

o Recommendations (e.g., through content-based approaches or collaborative filtering) are rated according to their relevance.

1. **Real-Time Updates**:

o We update the graph as users engage with the products (liking a new item, for example).

o Suggestions are dynamically modified in response to the changing graph

structure.

# When to use it

1. **Unstructured Data**:

o Situation: When working with data that doesn't neatly fit into columns and rows.

o A few examples are records, multimedia information, sensor readings, and posts on social media.

o Use of NoSQL: NoSQL databases support a variety of data formats by offering flexible schema design.

1. **High Write Throughput**:

o Situation: Applications that require fast data input and updates.

Examples include real-time analytics, event-driven systems, and IoT platforms.

o Use of NoSQL: NoSQL databases efficiently handle high write demands due its distributed architecture.

1. **Scalability Demands**:

o Scenario: Scaling horizontally becomes crucial as data volumes increase exponentially.

o As an illustration, consider content delivery networks, recommendation engines, and e-commerce platforms.

o Application of NoSQL: NoSQL databases ensure smooth scalability by distributing data among nodes.

1. **Graph-Based Relationships**:

o Scene: Representing intricate connections between different elements.

o As an illustration, consider social media, fraud detection, and recommendation engines.

o NoSQL usage: Neo4j and other graph databases are excellent in expressing and querying interconnected data.

1. **Time-Series Data**:

o Situation: Managing timestamp-based data points.

o Server logs, financial market tickers, and sensor data are a few examples.

o Application of NoSQL: Time-series databases, such as InfluxDB, are optimised for effective time-stamped data storage and retrieval.

1. **Polyglot Persistence**:

o Situation: When not every use case can be satisfied by a single data model.

o An illustration would be to combine key-value access, graph queries, and document storage.

o NoSQL usage: Multi-model databases provide flexibility without compromising efficiency.

In conclusion, NoSQL databases perform exceptionally well in situations where conventional relational databases struggle, such as managing complex relationships, supporting a variety of data kinds, or scaling horizontally. Select the best NoSQL solution for your particular domain and data types.

References

**NoSQL Referential Data**:

* Stack Overflow. [“NoSQL Referential Data.”](https://stackoverflow.com/questions/7591943/nosql-referential-data)