# **Big Data - Week 01 Report**

**Li weixuan wuzhenheng**

## **1. Real - World Database Examples Research**

### **a. Chosen Scenarios**

Our group has chosen the following scenarios for research:

* ****eCommerce****: Due to the complexity of organizing products, pricing, customer information, and purchase history.
* ****Healthcare****: Because of the need to manage extensive and sensitive patient data.
* ****Travel Reservation Systems****: As it involves maintaining a large amount of diverse information such as passenger details, flight schedules, and booking status.

### **b. Recommended Database Solutions**

****eCommerce****:

* + ****Database Type****: A combination of Relational (for structured customer and product information) and Document - Oriented NoSQL (for storing unstructured product descriptions and customer reviews) would be suitable.
  + ****Database Software Tools****: MySQL for the relational part and MongoDB for the NoSQL part.
  + ****Justification****:
    - ****Data Model****: The structured data like customer details and product pricing fits well in a relational model. However, unstructured data such as product reviews and detailed descriptions are better managed in a document - oriented database.
    - ****Scalability****: As the business grows, both MySQL and MongoDB offer good scalability options. MySQL can be scaled vertically, and MongoDB can be scaled horizontally.
    - ****Data Consistency****: ACID compliance in MySQL ensures data integrity for critical transactions like order processing. MongoDB's eventual consistency is acceptable for non - critical data like product reviews.
    - ****Performance****: Indexing in MySQL provides fast read and write operations for structured data. MongoDB's document - based storage allows for quick access to related data.
    - ****Security****: MySQL offers access control and encryption for sensitive customer data. MongoDB has security features to protect the unstructured data.
    - ****Cost****: Open - source options like MySQL and MongoDB reduce upfront and ongoing costs.
    - ****Community Support****: Both have large and active communities, providing ample resources and support.

****Healthcare****:

* + ****Database Type****: Relational database is the most appropriate due to the need for strict data integrity and complex querying of patient records.
  + ****Database Software Tools****: PostgreSQL.
  + ****Justification****:
    - ****Data Model****: Patient data is highly structured, including details such as medical history, test results, and treatment plans. A relational model allows for a clear and organized data structure.
    - ****Scalability****: PostgreSQL can be scaled both horizontally and vertically to handle the growing volume of patient data.
    - ****Data Consistency****: ACID compliance is crucial in healthcare to ensure the accuracy and consistency of patient information.
    - ****Performance****: Indexing and query optimization in PostgreSQL enable efficient access to patient data.
    - ****Security****: It offers strong security features such as access control, encryption, and auditing to protect sensitive patient data.
    - ****Cost****: Considering the need for a reliable and secure database, the cost of PostgreSQL is reasonable, especially with its open - source nature.
    - ****Community Support****: There is a large and active community of developers and healthcare IT professionals who contribute to the improvement and support of PostgreSQL in healthcare applications.

****Travel Reservation Systems****:

* + ****Database Type****: A hybrid approach with a Relational database for core transactional data (like passenger details and booking status) and a Graph database for representing complex relationships (such as flight routes and connections).
  + ****Database Software Tools****: Oracle Database for the relational part and Neo4j for the graph part.
  + ****Justification****:
    - ****Data Model****: The structured nature of passenger information and booking details is well - suited for a relational model. However, the complex network of flight routes, connections, and alternative routes is better represented using a graph database.
    - ****Scalability****: Oracle Database can handle large transaction volumes and can be scaled vertically. Neo4j is designed for handling complex graph - based data and can be scaled horizontally.
    - ****Data Consistency****: ACID compliance in Oracle Database ensures the integrity of booking transactions. Neo4j's graph - based model ensures consistency in representing network relationships.
    - ****Performance****: Indexing in Oracle Database provides fast access to passenger and booking data. Neo4j's graph traversal algorithms offer efficient querying of route - related data.
    - ****Security****: Oracle Database has robust security features for protecting passenger data. Neo4j also offers security measures for the graph data.
    - ****Cost****: While Oracle Database has licensing costs, its reliability and feature - set justify the expense for a critical application like travel reservations. Neo4j's cost is reasonable considering its specialized capabilities.
    - ****Community Support****: Both have active communities that provide support, best practices, and solutions for their respective domains.

## **2. Relational vs NoSQL Databases Comparison**

### **a. Comparison Table of Features**

| **Feature** | **Relational Databases** | **NoSQL Databases** |
| --- | --- | --- |
| Definitions | Store data in tables with predefined schemas and relationships. | Do not rely on a fixed schema and can handle various data models. |
| Benefits | Strong data consistency, well - established for complex queries. | High scalability, flexible schema for handling unstructured data. |
| Limitations | Schema rigidity can be a challenge for evolving data. | May lack strong consistency in some cases. |
| Examples of Software | MySQL, PostgreSQL. | MongoDB, Cassandra. |
| Use Cases | Enterprise applications with strict data integrity requirements. | Big data applications, social media platforms. |

### **b. Detailed Comparison Table**

| **Feature** | **Relational Databases** | **NoSQL Databases** |
| --- | --- | --- |
| Database Structures - Type of data and how it is stored | Data is stored in tables with rows and columns, following a predefined schema. | Can store data in various formats such as key - value pairs, documents, graphs, etc. |
| Data Storage - volumes of data | Can handle large amounts of data but may face challenges with extremely high - volume unstructured data. | Designed to handle massive volumes of unstructured and semi - structured data. |
| Do they support ACID transactions (Atomicity, Consistency, Isolation, Durability)? | Yes, ensuring high data integrity. | Some NoSQL databases support ACID, while others offer eventual consistency. |
| Is Normalisation supported? | Yes, to reduce data redundancy. | Not applicable in the traditional sense as there is no fixed schema. |
| Integrity constraints; Data Accuracy | Strong integrity constraints are enforced through the schema. | Data accuracy is maintained based on the specific NoSQL model used. |
| Scalability; horizontal and vertical scaling | Can be scaled both horizontally and vertically but may have limitations. | Highly scalable, especially horizontally. |
| Simplicity: ease of use, support available | Well - understood and widely used, with a large community for support. | Varying in ease of use depending on the specific NoSQL type, but with growing communities. |
| Complexity Cost | Can be complex to design and maintain, with costs associated with hardware and software. | May have lower upfront costs but could have higher costs for specialized skills. |
| Reliability | Reliable with established backup and recovery mechanisms. | Reliability depends on the specific NoSQL implementation. |
| Schema Flexibility | Schema is fixed, but can be modified, which can be costly. | Highly flexible schema that can adapt to changing data requirements. |
| Performance; read and write | Good for complex queries but may have slower write speeds for large datasets. | Fast write speeds for unstructured data, but query performance can vary. |
| Storage Requirements | Requires a defined storage structure based on the schema. | Can adapt storage requirements based on the data. |

## **3. Presentation**

### **a. Slides Information**

* ****Relational Databases****:
  + ****Benefit****: Strong data consistency and integrity.
  + ****Limitation****: Rigid schema that can be difficult to change.
  + ****Features Comparison****: Supports normalization vs NoSQL's flexible schema. ACID compliance vs eventual consistency in some NoSQL databases.
  + ****Use Case****: Ideal for financial applications where accurate and consistent data is crucial.
* ****NoSQL Databases****:
  + ****Benefit****: High scalability and ability to handle unstructured data.
  + ****Limitation****: May not provide the same level of data consistency as relational databases.
  + ****Features Comparison****: Stores data in various formats (e.g., documents, key - value) vs relational's tabular format. Horizontal scalability vs vertical scalability in some cases.
  + ****Use Case****: Suitable for social media platforms that deal with large volumes of unstructured user - generated content.