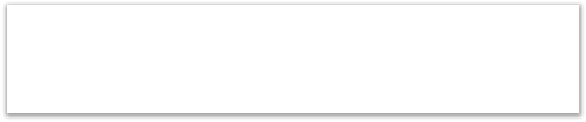


MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION (MUMBAI)



**CASE-STUDY REPORT**

“Understanding Databases”

***This*** ***Case study developed under the,***

***Program Name: Computer Enginnering Semester:4K***

***Course Title: Java Programming (JPR) CourseCode:314317***

**Submitted by,**

Name of Member:Nakil S.S. EnrollNo:24212620153

Name of Member:Kadabagaoker S.S. EnrollNo:23212620141

Name of Member:Mujawar M.S. EnrollNo:24212620152

**Under the guidance of,**

( Mr. Pawar V.D. )



Academic Year 2024-2025

**Certificate**

This is to certify that the following group of students submitted Case Study Report entitled **“Understanding Databases”** of  **4k** Semester of Diploma in **Computer Engineering** has completed the Case Study satisfactorily in course **Java Programming (JPR) (**course code:**314317**) for the academic Year **2024 to 2025** as prescribed in the curriculum.

**Submitted by**

Name of Member:Nakil S.S. EnrollNo:24212620153

Name of Member:Kadabagaoker S.S. EnrollNo:23212620141

Name of Member:Mujawar M.S. EnrollNo:24212620152

....…………………….. ………………..

**Case Study Guide H.O.D.**

Department Seal

**S.V.S.M.D’s**

**Kai. Kalyanrao (Balasaheb) Ingale Polytechnic, Akkalkot**

(Affiliated to MSBTE MUMBAI)

**2024- 2025**

---:: INDEX ::---

|  |  |  |
| --- | --- | --- |
| Sr. No. | Name of Topic | Page No. |
| 1.0 | Introduction |  |
| 2.0 | Types of Databases |  |
| 3.0 | Detailed Info About It |  |
| 4.0 | Easy To Use 2-3 Database |  |
| 5.0 | Installation Of a Database |  |
| 6.0 | Basic syntax |  |
| 7.0 | Conclusion |  |
| 8.0 | References |  |

**“Case-Study Title : Understanding Database”**

**1.0 Introduction**

In the digital era, data is the most valuable asset for individuals, businesses, and governments. As the volume of data continues to grow rapidly, the need to store, manage, retrieve, and analyze it efficiently becomes critical. This is where **databases** come into play.

A **database** is an organized collection of structured or unstructured data that can be accessed, modified, and managed electronically. Whether it's an e-commerce platform tracking customer orders, a social media app managing user profiles, or a banking system handling transactions, databases are at the core of almost every digital application.

This case study aims to provide a clear understanding of different types of databases, explore easy-to-use database systems, guide on installation, cover basic syntax, and introduce essential database programming concepts. It serves as a foundational guide for students, developers, and IT professionals who want to enhance their knowledge in the field of database management.

1. A database is a structured collection of data stored electronically for easy access, management, and updating.
2. Databases are essential in almost all fields including banking, education, e-commerce, healthcare, and social media.
3. They allow for efficient data storage, retrieval, modification, and security of information.
4. As data generation grows rapidly in the digital world, the role of databases becomes increasingly important.

* **This case study covers:**
* Different types of databases
* Commonly used free database systems
* How to install and use them
* Basic syntax and commands
* An introduction to PL/SQL programming

1. The goal is to help learners and professionals gain a clear understanding of how databases work and how to use them effectively.

**2.0 Types of Databases**

1. **Relational Database (RDBMS)**

➡ **Structure:** Tables with rows and columns

➡ **Example:** MySQL, PostgreSQL, Oracle, Microsoft SQL Server

**🔹 Key Features:**

* Uses Structured Query Language (SQL)
* Data stored in relations (tables)
* Supports constraints, keys, and normalization
* ACID (Atomicity, Consistency, Isolation, Durability) compliant
* Best for: Banking, ERP systems, websites, where data integrity is crucial.

1. **NoSQL Database**

➡ **Structure:** Non-tabular; varies by type

➡ **Example:** MongoDB, Cassandra, Redis, CouchDB

**🔹 Subtypes:**

* Document-oriented (e.g., MongoDB)
* Key-Value Stores (e.g., Redis)
* Column-Family (e.g., Apache Cassandra)
* Graph-Based (e.g., Neo4j)
* Best for: Big data, real-time applications, unstructured data, scalability.

1. **Object-Oriented Database**

➡ **Structure:** Stores data as objects, like in object-oriented programming

➡ **Example**: db4o, ObjectDB

**🔹 Key Features:**

* Stores class objects with attributes and methods
* Inheritance and encapsulation supported
* Best for: Complex applications where OOP principles need to be preserved in storage.

1. **Hierarchical Database**

➡ **Structure:** Tree-like (parent-child relationship)

➡ **Example:** IBM’s IMS (Information Management System)

**🔹 Key Features:**

* Each child has one parent (1:N relationship)
* Fast data access but limited flexibility
* Best for: File systems, legacy systems.

1. **Network Database**

➡ **Structure:** Graph-based (many-to-many relationships)

➡ **Example:** Integrated Data Store (IDS), IDMS

**🔹 Key Features:**

* More flexible than hierarchical
* Complex relationships supported
* Best for: Telecom, transport systems, and complex networks.

1. **Distributed Database**

➡ **Structure:** Data spread across multiple locations (servers or nodes)

➡ **Example:** Apache Cassandra, Google Spanner, Amazon DynamoDB

**🔹 Key Features:**

* Improves performance and availability
* Replication and consistency models applied
* Best for: Global applications, cloud environments.

1. **Cloud Database**

➡ **Structure:** Any type of DB hosted on the cloud

➡ **Example:** Amazon RDS, Google Firebase, Microsoft Azure SQL

**🔹 Key Features:**

* Scalable, flexible, and managed
* Pay-as-you-go model
* Easy integration with cloud apps
* Best for: SaaS applications, startups, mobile apps.

1. **Graph Database**

➡ **Structure:** Nodes and relationships (edges)

➡ **Example:** Neo4j, ArangoDB

**🔹 Key Features:**

* Ideal for connected data (social networks, recommendations)
* Uses graph theory
* Best for: Social networks, fraud detection, recommendation engines.

**3.0 Detailed Info About It**

Databases are essential tools for managing data in structured and unstructured forms. Each type of database has its unique design, features, and ideal use cases, making it important to understand their differences and advantages.

* **Relational Databases (RDBMS)** are the most widely used type. They store data in tables consisting of rows and columns, where each table is related to others through keys. These databases use Structured Query Language (SQL) for data management, ensuring high data integrity and consistency. Examples include MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. They are suitable for applications where data relationships and transactional accuracy are important, such as banking systems or ERP software.
* **NoSQL Databases** were developed to handle large volumes of unstructured or semi-structured data. They provide flexible schemas and are optimized for scalability and performance. Unlike relational databases, NoSQL systems do not rely on SQL. They are divided into several categories: document-based (e.g., MongoDB), key-value stores (e.g., Redis), column-family stores (e.g., Apache Cassandra), and graph databases (e.g., Neo4j). NoSQL is commonly used in big data, content management systems, and real-time web applications.
* **Object-Oriented Databases** store data as objects, similar to the way data is represented in object-oriented programming languages. They support object features like inheritance and encapsulation, which make them suitable for applications requiring tight integration between the database and programming logic. Examples include ObjectDB and db4o.
* **Hierarchical Databases** organize data in a tree-like structure where each child record has only one parent. This model is efficient for applications with clear hierarchical relationships, such as organizational charts or file systems. However, it is less flexible for complex relationships. IBM’s IMS is a well-known example.
* **Network Databases** improve on hierarchical models by allowing many-to-many relationships between entities, forming a graph-like structure. They provide more flexibility in data modeling and are used in industries like telecommunications, where complex data relationships are common.
* **Distributed Databases** store data across multiple physical locations, which could be on different servers or geographical areas. They ensure redundancy and fault tolerance, making them suitable for large-scale systems. Popular distributed systems include Apache Cassandra and Google Spanner.
* **Cloud Databases** are hosted on cloud platforms and accessed via the internet. They provide scalable storage, automatic backups, and high availability, making them ideal for modern web and mobile applications. Examples include Firebase, Amazon RDS, and Google Cloud SQL.
* **Graph Databases** specialize in storing and analyzing relationships between data. They represent data as nodes (entities) and edges (connections), which is ideal for applications like social networks, recommendation engines, and fraud detection systems. Neo4j is a leading graph database in this category.

Understanding these various database types helps in selecting the right tool for specific data needs and system requirements.

**4.0 Easy To Use 2-3 Database**

For beginners and professionals alike, certain databases stand out due to their ease of installation, user-friendly interfaces, strong community support, and extensive documentation. Among the many available, **MySQL**, **PostgreSQL**, and **MongoDB** are three of the most accessible and widely used database systems today.

1. **MySQL – (Relational Database)**

**Overview:**

MySQL is an open-source, SQL-based relational database management system developed by Oracle Corporation. It is one of the most popular databases used in web applications, especially in the LAMP stack (Linux, Apache, MySQL, PHP/Python/Perl).

**Key Features:**

* Uses standard SQL syntax.
* Supports ACID properties and transactions.
* Cross-platform and open-source.
* Widely used in web development (WordPress, Joomla, Drupal).
* Compatible with tools like phpMyAdmin (a web-based UI for managing MySQL databases easily).

**Pros:**

* Easy to install and configure.
* Huge community support and documentation.
* High performance and speed for read-heavy applications.

**Use Cases:**

* E-commerce websites
* Blogs and content management systems
* Small-to-medium scale enterprise applications

1. **PostgreSQL – (Advanced Relational Database)**

**Overview:**

PostgreSQL, often referred to as “Postgres,” is a powerful, open-source object-relational database system with more than 30 years of active development. It’s known for its standards compliance, performance, and extensibility.

**Key Features:**

* Advanced SQL support including full joins, views, stored procedures, and custom functions.
* Supports JSON, arrays, and user-defined data types.
* Highly extensible with add-ons and plug-ins.
* ACID-compliant and supports MVCC (multi-version concurrency control).
* Compatible with tools like pgAdmin for graphical user interaction.

**Pros:**

* More robust and scalable than MySQL.
* Excellent support for complex queries and large datasets.
* Extensible architecture (can define your own data types, operators, etc.)

**Use Cases:**

* Enterprise-level applications
* GIS and spatial data systems
* Financial and government databases

1. **MongoDB – (NoSQL Document Database)**

**Overview:**

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It is ideal for applications that require a dynamic schema and high-speed data operations. MongoDB is particularly useful in modern application development due to its flexibility and scalability.

**Key Features:**

* Schema-less data storage (no need to define a structure beforehand).
* Uses BSON (Binary JSON) for efficient storage.
* Built-in replication, high availability, and horizontal scaling.
* Powerful query language with support for indexing and aggregation.
* Comes with a web-based GUI called MongoDB Compass.

**Pros:**

* Extremely flexible – perfect for rapid development.
* Scales easily across multiple servers.
* Great integration with JavaScript-based stacks like Node.js.

**Use Cases:**

* Real-time analytics
* Content management systems
* IoT applications
* Mobile and web applications

### ****Summary Comparison Table****

| **Feature** | **MySQL** | **PostgreSQL** | **MongoDB** |
| --- | --- | --- | --- |
| Type | Relational (SQL) | Relational (Advanced SQL) | NoSQL (Document-based) |
| Structure | Tables | Tables + JSON support | JSON-like Documents |
| Best For | Web apps, CMS | Complex queries, analytics | Scalable web/mobile apps |
| GUI Tool | phpMyAdmin | pgAdmin | MongoDB Compass |
| Schema | Fixed | Fixed + JSON | Schema-less |
| Learning Curve | Easy | Medium | Easy |

**5.0 Installation Of a Database**

This section explains how to install three commonly used databases—MySQL, PostgreSQL, and MongoDB—on a Windows system. The process includes step-by-step instructions with links to official sources for easy and secure downloading.

1. **Installing MySQL on Windows**

**Step-by-step:**

1. Download the MySQL Installer:

* Visit: [https://dev.mysql.com/downloads/installer/](https://dev.mysql.com/downloads/installer/" \t "_new)
* Choose either the Web Installer (small download) or the Full Installer (includes all components).

1. Run the Installer:

* Launch the downloaded .msi file.
* Choose "Developer Default" or "Server only" based on your needs.

1. Set Configuration:

* Select a server configuration type (usually "Standalone MySQL Server").
* Set the root password (important for security).
* Optionally create additional user accounts.

1. Start MySQL Server:

* The service will start automatically after installation.
* Use MySQL Workbench (installed with the setup) to interact with your database.

1. Verify Installation:

* Open MySQL Command Line Client or Workbench.
* Run a simple command:

1. **Installing PostgreSQL on Windows**

**Step-by-step:**

1. Download the Installer:

* Go to: https://www.postgresql.org/download/windows/
* Download the installer from EnterpriseDB.

1. Run the Installer:

* Double-click the .exe file to launch the installer.
* Choose installation directory, and proceed with default components.

1. Set User & Password:

* Set a password for the "postgres" user (admin user).
* Choose port (default is 5432) and locale.

1. Finish Setup:

* Complete installation and launch pgAdmin (graphical admin tool).

1. Verify Installation:

* Open pgAdmin, connect to server with the credentials.

1. Run a test query:

* SELECT version();

1. **Installing MongoDB on Windows**

**Step-by-step:**

1. Download MongoDB Installer:

* Visit: https://www.mongodb.com/try/download/community
* Choose MSI installer for your Windows version.

1. Run the Installer:

* Double-click the .msi file and follow the wizard.
* Choose "Complete" installation.
* Tick the option to Install MongoDB as a Service.

1. Install MongoDB Compass (Optional):

* The installer gives the option to install MongoDB Compass, a GUI for MongoDB.

1. Verify Installation:

Open Command Prompt and type:

* mongod --version

Start MongoDB Service (if needed):

* net start MongoDB

**6.0 Basic syntax**

### ****SQL Syntax (MySQL & PostgreSQL)****

#### ➤ ****Create a Database****

sql

CREATE DATABASE mydatabase;

#### ➤ ****Use a Database****

sql

USE mydatabase;

#### ➤ ****Create a Table****

sql

CREATE TABLE students (id INT PRIMARY KEY, name VARCHAR(100),age INT);

#### ➤ ****Insert Data****

sql

INSERT INTO students (id, name, age)VALUES (1, 'Alice', 22);

#### ➤ ****Retrieve Data****

sql

SELECT \* FROM students ;

#### ➤ ****Update Data****

sql

UPDATE students SET age = 23WHERE id = 1;

#### ➤ ****Delete Data****

sql

DELETE FROM students WHERE id = 1;

#### ➤ ****Drop a Table****

sql

DROP TABLE students;

### ****MongoDB Syntax (NoSQL)****

MongoDB uses **JavaScript-like syntax** and does not use tables. It works with **collections** and **documents**.

#### ➤ ****Use/Create a Database****

javascript

use mydatabase

#### ➤ ****Create/Insert into Collection****

javascript

db.students.insertOne({ id: 1, name: "Alice", age: 22 });

#### ➤ ****Find Data****

javascript

db.students.find(); // List all

db.students.find({ name: "Alice" }); // Filtered

#### ➤ ****Update Document****

javascript

db.students.updateOne({ id: 1 },{ $set: { age: 23 } });

#### ➤ ****Delete Document****

javascript

db.students.deleteOne({ id: 1 });

#### ➤ ****Drop Collection****

javascript

db.students.drop();

**7.0 Conclusion**

Databases play a crucial role in the management, storage, and retrieval of data across all types of applications, from simple websites to complex enterprise systems. This case study explored various types of databases, including relational (like MySQL and PostgreSQL) and NoSQL (like MongoDB), each offering unique features suited for different use cases.

We discussed easy-to-use databases that are beginner-friendly yet powerful enough for advanced applications. Installation processes on Windows were outlined to help users get started practically. Understanding basic syntax—both SQL and NoSQL—equips learners with the foundational skills needed to interact with these systems effectively.

As technology continues to evolve, so does the need for efficient data management. Choosing the right database, understanding its structure, and learning how to work with it not only enhances application performance but also opens the door to better data-driven decision-making. Whether you're a student, developer, or tech enthusiast, gaining proficiency in databases is a vital step toward mastering the world of computing.

1. Databases are essential for storing, managing, and retrieving data in almost every software application.
2. There are different types of databases, such as Relational (SQL) and NoSQL, each serving unique purposes.
3. MySQL, PostgreSQL, and MongoDB are three user-friendly databases that are easy to install and widely used in the industry.
4. We covered how to install databases on Windows, including step-by-step instructions for all three major systems.
5. Understanding basic syntax for both SQL (used in MySQL/PostgreSQL) and NoSQL (used in MongoDB) helps in performing CRUD (Create, Read, Update, Delete) operations efficiently.
6. Mastering databases improves a developer’s ability to build dynamic and data-driven applications.
7. Learning and using databases is a vital skill for students, developers, and IT professionals in today’s tech-driven world.

**8.0 References**

**Websites :**

* https://www.mysql.com/ – Official website for MySQL documentation and downloads
* https://www.postgresql.org/ – PostgreSQL official site with tutorials and tools
* https://www.mongodb.com/ – MongoDB documentation, downloads, and Compass GUI
* https://www.w3schools.com/sql/ – Beginner-friendly SQL tutorials
* https://www.geeksforgeeks.org/dbms/ – Concepts, notes, and examples on database systems
* https://tutorialspoint.com/sql/ – Step-by-step SQL learning resource
* https://www.javatpoint.com/dbms-tutorial – Comprehensive DBMS notes and examples

**Books :**

**"Database System Concepts"** by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan

* Widely used in universities for in-depth understanding of DBMS principles.

**"SQL For Dummies"** by Allen G. Taylor

* Great for beginners who want a simple introduction to SQL.

**"Learning SQL"** by Alan Beaulieu

* A practical guide for mastering SQL through hands-on exercises.

**"MongoDB: The Definitive Guide"** by Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow

* Covers NoSQL concepts and how MongoDB works in modern applications.

**"PostgreSQL: Up and Running"** by Regina Obe and Leo Hsu

* A solid resource for getting started with PostgreSQL quickly.