Database Management System
Mid-Term Report

Dola Surya Sai

June 24,2024

Contents

1 Introduction to Databases and DBMS

1.1 Introduction

In the modern digital era, data is an invaluable asset, driving decisions and operations across various sectors. Databases and Database Management Systems (DBMS) are critical in organizing, storing, and managing this data efficiently. This essay provides an overview of databases and DBMS, their importance, types, and key functionalities.

1.2 What is a Database?

A database is a structured collection of data that allows for efficient retrieval, insertion, and deletion of information. Databases are designed to manage large volumes of information and provide mechanisms to ensure data consistency, integrity, and security.

1.2.1 Key Characteristics

- **Structured Data:** Databases store data in a structured format, typically using tables (in relational databases), documents (in document databases), or other forms of data models.
- **Efficient Data Management:** They provide efficient methods for storing, retrieving, and manipulating data.
- **Data Integrity and Consistency:** Databases enforce rules to ensure the accuracy and consistency of data.

1.3 What is a DBMS?

A Database Management System (DBMS) is a software system that interacts with the database, users, and applications to capture and analyze data. It provides tools to define, create, maintain, and control access to the database.

1.3.1 Key Functions of DBMS

- **Data Definition:** Allows users to define the structure of the data, such as creating tables, defining relationships, and setting constraints.
- **Data Manipulation:** Provides tools to insert, update, delete, and retrieve data from the database.
- **Data Security:** Ensures that only authorized users have access to the data, protecting it from unauthorized access and breaches.
- **Data Integrity:** Maintains data accuracy and consistency through integrity constraints and transactions.
- **Data Backup and Recovery:** Provides mechanisms for data backup and recovery in case of system failures or data corruption.

1.4 Types of Databases

There are various types of databases, each designed to meet specific needs and use cases. The main types include:

1.4.1 Relational Databases

- **Structure:** Data is stored in tables with rows and columns.
- **Examples:** MySQL, PostgreSQL, Oracle.
- **Advantages:** Highly flexible, supports complex queries, ensures data integrity.
- **Use Cases:** Widely used in business applications, financial systems, and web applications.

1.4.2 NoSQL Databases

- **Structure:** Can be document-based, key-value pairs, column-family stores, or graph databases.
- **Examples:** MongoDB (document), Redis (key-value), Cassandra (column-family), Neo4j (graph).
- **Advantages:** Flexible schema, scalable, handles large volumes of unstructured data.

- **Use Cases:** Real-time web applications, big data analytics, social networks.

1.4.3 Object-Oriented Databases

- **Structure:** Integrates with object-oriented programming languages, storing data as objects.
- **Examples:** ObjectDB, db4o.
- **Advantages:** Seamless integration with object-oriented applications, supports complex data types.
- **Use Cases:** CAD/CAM, complex data management applications.

2 Database Models: An Overview from "Fundamentals of Database Systems" by Elmasri and Navathe

2.1 Introduction

Database models are fundamental to understanding how data is structured, stored, and managed within a database management system (DBMS). Chapters 3 and 4 of "Fundamentals of Database Systems" by Ramez Elmasri and Shamkant B. Navathe delve into various database models, exploring their structures, advantages, and applications. This essay provides an overview of the key database models discussed in these chapters, including the hierarchical, network, relational, and object-oriented models.

2.2 Hierarchical Model

The hierarchical model organizes data in a tree-like structure where each record has a single parent, forming a parent-child relationship. This model is particularly effective for representing data with a clear hierarchy, such as organizational structures or file systems.

2.2.1 Structure and Characteristics

- **Tree Structure:** The hierarchical model arranges data in a tree with nodes representing records and edges representing relationships.
- **Parent-Child Relationships:** Each child node has one parent node, but a parent can have multiple child nodes.
- **Data Access:** Data retrieval involves navigating through the tree from the root node to the desired child nodes.

2.2.2 Advantages and Disadvantages

- **Advantages:** Simple to design and efficient for queries that follow the hierarchical path.
- **Disadvantages:** Limited flexibility due to the strict parent-child relationship, making it challenging to represent many-to-many relationships.

2.3 Network Model

The network model extends the hierarchical model by allowing more complex relationships through a graph structure. It supports many-to-many relationships, providing greater flexibility in representing real-world scenarios.

2.3.1 Structure and Characteristics

- **Graph Structure:** Uses nodes and edges to represent entities and relationships, respectively.
- **Complex Relationships:** Supports many-to-many relationships, enabling more intricate data connections.
- **Data Access:** Data retrieval involves traversing the graph, which can handle more complex queries compared to the hierarchical model.

2.3.2 Advantages and Disadvantages

- **Advantages:** More flexible and capable of representing complex relationships. Efficient for certain types of queries.
- **Disadvantages:** More complex to design and maintain, which can lead to increased development time and cost.

2.4 Relational Model

The relational model, introduced by E.F. Codd, is the most widely used database model. It organizes data into tables (relations) consisting of rows (tuples) and columns (attributes). This model is highly flexible and supports powerful query languages like SQL.

2.4.1 Structure and Characteristics

- **Table Structure:** Data is stored in tables, each consisting of rows and columns.
- **Primary and Foreign Keys:** Tables are linked using primary keys (unique identifiers for rows) and foreign keys (references to primary keys in other tables).

- **Data Integrity:** Enforces data integrity and consistency through constraints and normalization.

2.4.2 Advantages and Disadvantages

- **Advantages:** Highly flexible, supports complex queries, and ensures data integrity. It is the standard for most database applications.
- **Disadvantages:** Can be less efficient for very large datasets and certain types of queries compared to other models.

2.5 Object-Oriented Model

The object-oriented model integrates database concepts with object-oriented programming principles. It stores data as objects, similar to how data is handled in object-oriented programming languages.

2.5.1 Structure and Characteristics

- **Object Structure:** Data is stored as objects, which contain both data (attributes) and methods (functions).
- **Inheritance and Encapsulation:** Supports inheritance, encapsulation, and polymorphism, allowing for complex data structures and relationships.
- **Class Hierarchies:** Objects are instances of classes, which can inherit properties and methods from other classes.

2.5.2 Advantages and Disadvantages

- **Advantages:** Supports complex data types and relationships. Seamless integration with object-oriented programming languages.
- **Disadvantages:** Can be more complex to design and maintain. Less mature tooling and support compared to the relational model.

2.6 Conclusion

Database models play a crucial role in the design and implementation of databases. Each model offers unique advantages and is suited to different types of applications and data structures. The hierarchical model is simple and efficient for hierarchical data, while the network model provides greater flexibility for complex relationships. The relational model is highly versatile and widely used, making it the standard for most database applications. The object-oriented model, on the other hand, offers powerful features for handling complex data and relationships, integrating seamlessly with object-oriented programming languages. Understanding these models and their characteristics is essential for designing effective and efficient database systems.

3 Database Design

Database design refers to the process of producing a detailed data model of a database. It involves defining the structure, organization, and relationships of the data stored in the database. The goal of database design is to produce a design that supports the storage, manipulation, and retrieval of data efficiently and effectively.

3.1 Key Aspects of Database Design

- **Requirements Analysis:** Understanding and documenting the requirements of the database system. This involves gathering information about what data needs to be stored, how it will be used, and what operations will be performed on it.
- **Conceptual Design:** Creating a high-level conceptual model of the database, often using entity-relationship diagrams (ER diagrams). This step focuses on identifying entities (objects or things of interest), their attributes (properties or characteristics), and the relationships between entities.
- **Logical Design:** Transforming the conceptual model into a logical model that can be implemented in a database management system (DBMS). This involves translating the ER diagrams into tables, specifying attributes and data types, defining primary and foreign keys, and ensuring normalization to reduce redundancy and improve data integrity.
- **Physical Design:** Defining the physical storage and access methods for the database on the target DBMS platform. This includes decisions on data storage structures (e.g., tablespaces, indexes), optimization of queries and transactions, and considerations for performance tuning and scalability.
- **Implementation and Testing:** Creating the actual database schema based on the design, loading initial data, and testing the database to ensure it meets the specified requirements and performs efficiently.
- **Maintenance and Evolution:** Continuously monitoring and maintaining the database to ensure data integrity, security, and performance. Database design should also accommodate changes over time, such as new requirements, updates to data models, and optimizations.

4 SQL Basics

Structured Query Language (SQL) is a fundamental tool for managing and manipulating relational databases. It comprises two main categories: Data Definition Language (DDL) and Data Manipulation Language (DML), each serving distinct yet interrelated purposes in database management.

4.1 Data Definition Language (DDL)

DDL commands are used to define, modify, and delete database objects such as tables, indexes, and views. These commands provide the structural framework of the database.

4.1.1 CREATE

```sql
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Username VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE
);
```

Here, INT and VARCHAR are datatypes.Also the number in paranthesis is the maximum of characters a string can contain. And NOT NULL is representing the Username should not be null. UNIQUE is to represent the email should be unique for different entries.

4.1.2 ALTER

```sql
ALTER TABLE Users
ADD DateOfBirth DATE;
```

Here, by using ALTER we can add, delete, modify columns.And we can also change the data type of a column, and manage constraints and indexes.

4.1.3 DROP

```sql
DROP TABLE Users;
```

This is to drop a table.

4.2 Data Manipulation Language (DML)

DML commands are used to manage data within the database, including querying, inserting, updating, and deleting records.

4.2.1 SELECT

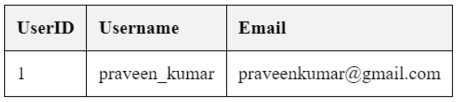
```sql
SELECT Username
FROM Users;
```

By using we can select contents of a table. We can select a particular column or we can select some columns or we can select all columns and we can manipulate using this.

4.2.2 INSERT

```sql
INSERT INTO Users (UserID, Username, Email)
VALUES (1, 'praveen_kumar', 'praveenkumar@gmail.com');
```

By this, we can insert values into that tables. After inserting, it looks like Fig. 1.



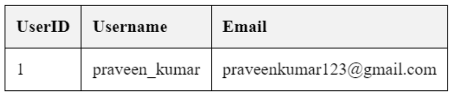
UserID	Username	Email
1	praveen_kumar	praveenkumar@gmail.com

Figure 1: INSERTING

4.2.3 UPDATE

```sql
UPDATE Users
SET Email = 'praveenkumar123@gmail.com'
WHERE UserID = 1;
```

Here, SET is to give a particular value to that variable and WHERE is like a condition that says when UserID equals to 1 change the email of that UserID to that particular value. And it looks like Fig. 2



UserID	Username	Email
1	praveen_kumar	praveenkumar123@gmail.com

Figure 2: UPDATING

4.2.4 DELETE

```sql
DELETE FROM Users
WHERE UserID = 1;
```

After deleting it will be empty.

4.3 Basic SQL Queries

SQL queries can be simple or complex, depending on the requirements. Here are some basic examples:

Selecting specific columns from a table:

```sql
    SELECT Username, Email
    FROM Users;

```

Filtering results using WHERE clause:

```sql
SELECT *
FROM Users
WHERE Username = 'john_doe';

```

Ordering results using ORDER BY:

```sql
SELECT *
FROM Users
ORDER BY Username ASC;

```

Joining tables using JOIN:

```sql
SELECT Orders.OrderID, Users.Username
FROM Orders
INNER JOIN Users ON Orders.UserID = Users.UserID;

```

SQL is a powerful language that enables efficient management and manipulation of data within relational databases. Understanding DDL and DML commands, as well as basic SQL queries, is essential for anyone working with databases, from developers designing database schemas to analysts extracting insights from data.

In conclusion, SQL's flexibility and robustness make it indispensable in the world of database management, providing both novice and experienced users with powerful tools to handle data effectively.

Database management system Revised PoA

Deadlines

- 4th July - Advanced covering Joins (INNER, OUTER, LEFT, RIGHT), Subqueries, Indexes, and Views.
- 8th July - Relational Database Theory
- 12th July - Transactions and Concurrency Control focusing on ACID properties,concurrency control techniques, and deadlock prevention and detection.
- 15th July - Final submission

Resources

- Fundamentals of database systems[Ramez Elmsari, Shamkant B.Navathe]