MANGALAYATAN UNIVERSITY
Learn Today to Lead Tomorrow
NAAC GRADE A+ Accredited University

BACHELOR OF COMPUTER APPLICATION (BCA)

CSB-1111 (Management Information System)

Block 1: Introduction to Management Information Systems

Unit 3: Database Management Systems

Structure	
3.0	Introduction
3.1	Objectives
3.2	Data and Its Origin
3.3	Data Types and Sources
3.4	Database Management System
3.4.1	Types of Databases
3.5	Relational Database Management System
3.6	Database Design
3.7	Database Normalization
3.8	Database Languages
3.9	Question and Answers
3.10	References

3.0 Introduction

In today's data-driven world, the ability to manage and utilize vast amounts of information is a critical asset for organizations across various sectors. Data is gathered from numerous sources and in various forms, making it essential to have robust systems in place for efficient data handling. This is where Database Management Systems (DBMS) come into play, providing structured and systematic ways to store, manage, and retrieve data. DBMS ensure data integrity, security, and accessibility, making them indispensable tools in the modern data landscape.

This chapter delves into the foundational aspects of data and its origins, categorizing different types and sources of data. It also explores the fundamental principles and types of DBMS, with a particular focus on Relational Database Management Systems (RDBMS), which are extensively used due to their efficiency in handling structured data. Additionally, we will cover essential concepts of database design and normalization, which are crucial for creating efficient, reliable, and scalable database systems.

Furthermore, we will examine various database languages, such as SQL, that are used to interact with and manage databases effectively. By understanding these core concepts, readers will be equipped with the knowledge necessary to design, implement, and maintain database systems, enabling them to harness the full potential of their data to support strategic organizational goals.

3.1 Objectives

By the end of this chapter, you should be able to:

- Describe what data is and identify its various origins.
- Explain the importance of data in decision-making processes.
- Differentiate between structured, unstructured, and semi-structured data.
- Identify and provide examples of different sources of data, including human-generated, machine-generated, and public data.
- Distinguish between various types of databases, including relational, NoSQL, and others.
- Understand the specific use cases and benefits of each type.

3.2 Data and Its Origin

Definition: The Latin word data is the plural of datum, "(thing) given", neuter past participle of dare, "to give".

The word "data" was first used in English in the 1640s. In 1946, the term "transmissible and storable computer information" was originally applied to the term "data". In 1954, the term "data processing" was coined.

Data refers to pieces of information that are collected, processed, and used for various purposes. It comes in many forms and originates from a wide variety of sources. Here's a comprehensive overview of data and its origins:

Data are commonly used to describe a set of discrete or continuous values that can describe quantity, quality, fact, statistics, or other fundamental units of meaning. They can also just be sequences of symbols that can be further understood technically. A single value in a set of data is called a datum. Typically, data are arranged into structures like tables, which can both be utilized as data in bigger structures and offer further context and meaning. Within a computer process, data can be utilized as variables. Concrete measures or abstract concepts can be represented by data. Statistics are widely employed in economics, science, and almost all other human organized activities. Price indices (such the consumer price index), unemployment, literacy, and census statistics are a few examples of data sets. In this sense, data are the unprocessed facts and numbers that can be used to derive meaningful information.

After being gathered through methods like measurement, observation, questioning, or analysis, data are usually expressed as numbers or characters that can then be processed further. Data gathered in an uncontrolled in-situ setting are referred to as field data. Data obtained during a well monitored scientific experiment are referred to as experimental data. Techniques including computation, logic, debate, presentation, visualization, and other post-analysis methods are used to analyze data. Raw data, also known as unprocessed data, are usually cleansed before analysis: Notable instrument or data entry problems are fixed, and outliers are eliminated.

The smallest bits of factual information that can serve as a foundation for computation, reasoning, or discussion are called data. Abstract concepts and tangible measurements, including but not limited to statistics, are examples of data. Information can be defined as data that is thematically related and provided in a pertinent context. Then, contextually related information might be referred to as intelligence or data insights. Knowledge is thus defined as the store of wisdom and intelligence that develops over time as a result of the synthesis of data into information. Something called "the new oil of the digital economy" has

been applied to data. In general, the term "data" refers to the fact that some knowledge or information already in existence is recorded or coded in a way that makes it easier to use or process.

3.3 Data Types and Sources

The phrases structured, unstructured, and semi-structured data are frequently brought up when we discuss data and analytics. These three categories of data are currently considered relevant for all commercial applications. Traditional systems and reporting still rely on structured data, which has been around for a while.

However, as big data has become more popular, the amount of semi-structured and unstructured data sources has rapidly increased in recent years. Because of this, an increasing number of companies are attempting to include all three types of data in their business analytics and business intelligence to further their efforts.

- Comparing Semi-Structured, Unstructured, and Structured Data
- Structured, Semi-Structured, and Unstructured Data Distinctions

Structured Data: Information that has been formatted and converted into a precise data model is known as structured data. The unprocessed data is transferred into pre-made fields so that SQL may read and extract it with ease. Tables with rows and columns seen in SQL relational databases are the ideal representation of organized data.

This data format's relational model makes use of memory since it reduces data redundancy. But it also implies that structured data is less flexible and more interdependent. Let's now examine some further instances of structured data.

Characteristics:

- Organized into rows and columns.
- Easily entered, stored, queried, and analyzed.
- Usually stored in relational databases like SQL.

Examples: Spreadsheets: A Microsoft Excel file with columns for customer names, addresses, phone numbers, and purchase amounts.

SQL Databases: A table in a MySQL database containing employee records with fields for employee ID, name, department, and salary.

Both people and machines produce this kind of data. Structured data from machines can be found in many different forms, including barcodes, weblog statistics, and quantity data from point-of-sale systems. Similarly, spreadsheets are a famous example of structured data created by people; almost everyone involved in data work has used one at some point in their careers. Structured data is easier to analyze than semi-structured and unstructured data because of its arrangement.

Semi-Structured Data: Your data sets might not always be organized or unstructured. Between structured and unstructured data, there is also semi-structured or partially structured data. One kind of data that has some specific and consistent properties is semi-structured data.

It is not constrained by a strict structure like relational databases require. Businesses manage semi-structured data more easily by using organizational features like semantic tags or metadata. It is still quite inconsistent and variable, though.

Characteristics:

- Contains tags or markers to separate data elements.

- More flexible than structured data but more organized than unstructured data.
- Often stored in formats like JSON, XML, or CSV.

Example: Delimited files are an example of data in a semi-structured format. It has components that allow the data to be divided into several hierarchies. Similar to this, digital photos are semi-structured even though they lack a predetermined framework due to a few structural characteristics.

- **JSON Files:** A JSON file containing product details such as **{"productID": 123, "name": "Laptop", "price": 799.99, "features": ["8GB RAM", "256GB SSD"]}**.
- **XML Files:** An XML file with employee data such as <employee><id>001</id><name>John Doe</name><department>HR</department></employee>.
- **Email Metadata:** The structure of email metadata like sender, recipient, timestamp, and subject.

A photo taken with a smartphone will include certain structured properties such as date and time stamp, device ID, and geotag. Once you've saved them, you can give photos labels like "dog" or "pet" to give them some structure. Because it possesses one or more classifying qualities, unstructured data can occasionally be categorized as semi-structured data.

Unstructured Data: Unstructured data is defined as data present in absolute raw form. This data is difficult to process due to its complex arrangement and formatting. Unstructured data includes social media posts, chats, satellite imagery, IoT sensor data, emails, and presentations. Unstructured data management takes this data to organize it in a logical, predefined manner in data storage. Natural language processing (NLP) tools help understand unstructured data that exists in a written format. In contrast, the meaning of structured data is data that follows predefined data models and is easy to analyze. Structured data examples would include alphabetically arranged names of customers and properly organized credit card numbers. After understanding the definition of unstructured data, let's look at some examples.

Characteristics:

- No predefined format or organization.
- More challenging to store and analyze.
- Often stored in data lakes or NoSQL databases.

Example: Unstructured data can be anything that's not in a specific format. This can be a paragraph from a book with relevant information or a web page. An example of unstructured data could also be Log files that are not easy to separate. Social media comments and posts are also unstructured.

- **Text Documents**: A collection of Word documents containing reports, articles, or meeting notes.
- **Emails**: Emails from various clients and stakeholders with different subjects, content, and attachments.
- **Social Media Posts**: Posts, comments, images, and videos shared on platforms like Twitter, Facebook, and Instagram.
- **Multimedia Files**: Photos, audio recordings, and videos, such as a folder of marketing videos or a database of stock images.

Businesses extract and store unstructured data in data warehouses (also called data lakes) for analysis. Unstructured data is qualitative, not quantitative, so it is mostly categorical and characteristic in nature. For instance, data from websites or social media can help predict future buying trends or determine the effectiveness of a marketing campaign. Another example of unstructured data analytics is detecting patterns in scam emails and chat, which can be useful for enterprises in monitoring policy compliance.

3.4 Database Management System

What is a database management system?

The system software used to create and administer databases is called a database management system (DBMS). End users can create, protect, read, update, and remove data in a database with the help of a DBMS. The most common kind of data management platform, the database management system (DBMS) functions simply as a conduit between users or application programs and databases, keeping data conveniently available and regularly organized.

Usage of DBMS:

Data is managed by the database management system (DBMS); its logical structure is specified by the database schema; and data access, locking, and modification are made possible by the database engine. In addition to providing concurrency, security, and data integrity, these three fundamental components also support standard data management practices. Numerous common database administration functions, such as change management, security, backup and recovery, and performance monitoring and tuning, are supported by the DBMS. Together with automatic rollbacks and restarts, logging, and auditing of database and application activities, the majority of database management systems also handle these tasks.

Multiple users from various places can get a centralized view of the data in a regulated manner thanks to the DBMS. Despite offering several views of a same database schema, a DBMS has the ability to restrict the data that end users see and how they view it. Because the DBMS manages all requests, end users and software applications are free from needing to comprehend where the data is physically housed or what kind of storage medium it sits on.

To shield users and applications from needing to know where data is stored or from worrying about changes to the physical structure of data, DBMSs can provide logical and physical data independence. Developers won't need to update programs just because the database has changed as long as applications utilize the application programming interface (API) for the database that the DBMS offers.

Advantages and Disadvantages of DBMS

Advantages of DBMS

1. **Data Redundancy and Inconsistency Reduction**: A DBMS minimizes data redundancy and inconsistency by ensuring that data is integrated and managed centrally, eliminating duplicate storage of the same data across different files. **Example**: In a university system, student information such as name, ID, and address is stored in a single place, reducing redundancy.
2. **Data Integrity**: Integrity constraints are enforced in a DBMS, ensuring the accuracy and consistency of data over its entire lifecycle. **Example**: Integrity constraints like primary keys ensure that no two rows have the same identifier in a student database.
3. **Data Security**: A DBMS provides a framework for enforcing user access controls, ensuring that only authorized users can access or modify data. **Example**: In a bank database, only authorized personnel can access sensitive customer information.
4. **Data Abstraction and Independence**: The DBMS abstracts the data, providing a conceptual view independent of the physical data storage details. **Example**: Users interact with high-level query languages like SQL, without needing to understand the underlying storage structures.
5. **Efficient Data Access**: DBMS uses sophisticated algorithms to store and retrieve data efficiently. **Example**: Indexing in databases allows for quick search and retrieval of data records.

6. **Concurrent Access and Crash Recovery**: A DBMS supports concurrent access, ensuring that multiple users can interact with the database simultaneously without conflicts, and it can recover from crashes to maintain data integrity. **Example**: In an online booking system, multiple users can book tickets simultaneously without issues.
7. **Backup and Recovery**: DBMS provides automated backup and recovery procedures to protect data against loss or corruption. **Example**: Regular backups ensure that data can be restored to a previous state in case of hardware failure.
8. **Data Consistency**: **Advantage**: The DBMS ensures that data modifications are consistently applied across the database, maintaining data integrity. **Example**: Transactions ensure that either all steps of a process are completed successfully or none are, maintaining consistency.

Disadvantages of DBMS

1. **Complexity**: DBMS systems are complex software systems that require specialized knowledge to install, configure, manage, and use. **Example**: Database administrators (DBAs) need to have in-depth knowledge of database management to effectively manage a DBMS.
2. **Cost**: Implementing a DBMS can be expensive due to the cost of software, hardware, and personnel. **Example**: Enterprise DBMS solutions like Oracle or SQL Server can be costly to license and maintain.
3. **Performance Overhead**: The abstraction and additional functionalities of a DBMS can introduce performance overhead compared to simpler file-based systems. **Example**: For small, simple applications, the performance overhead of a DBMS might not be justified compared to using a flat file system.
4. **Security Risks**: Centralizing data increases the risk that if unauthorized access is gained, a large amount of sensitive data could be compromised. **Example**: A security breach in a DBMS could expose all the stored data to attackers.
5. **Maintenance and Upgrades**: Regular maintenance, updates, and upgrades of a DBMS require significant effort and expertise. **Example**: Upgrading the DBMS software may involve downtime and complex migration processes.
6. **Backup Complexity**: Although DBMS systems support backup, the process can be complex and time-consuming, especially for large databases. **Example**: Ensuring that backups are consistent and complete requires careful planning and execution.

3.4.1 Types of Databases

There are different kinds of databases used to store different kinds of information:

1. **Centralized Database:** This specific type of database uses a centralized database system to store information. Retrieving saved data from many apps and locations is helpful to users. Users of these programs can safely access data thanks to an authentication scheme. A central library that keeps track of all the libraries in a school or college is an example of a centralized database.
2. **Distributed Database:** Instead of having a single central database system, data in distributed systems is dispersed over multiple organizational database systems. Communications links allow these database systems to communicate with one another. Accessing the data is made simpler for users via these connections. Distributed databases include Ignite, HBase, and Apache Cassandra.
3. **Relational Database:** This database's relational data model stores data in rows, often known as tuples, and columns, sometimes known as attributes, which come together to form a table (relation).

SQL is used in relational databases to store, manage, and safeguard data. E.F. Codd constructed the database in 1970. Each table in the database has a unique key that distinguishes its contents from those of other tables. SQL Server, Oracle, MySQL, and other relational databases are examples.

4. **NoSQL Database:** Not just in SQL, but in a wide variety of data sets, can be stored in a SQL database type. Since it doesn't store data in tables, it isn't a relational database. The desire for more contemporary applications led to its creation. In response, NoSQL developed a range of database solutions.
5. **Cloud Database:** A database that uses a cloud computing platform's virtual environment to store data. It gives users access to a range of cloud computing solutions (SaaS, PaaS, IaaS, etc.) for databases. Numerous cloud systems are accessible. But these are the top ones, as stated below:
    a. Microsoft Azure
    b. PhonixNAP
    c. Kamatera
    d. AWS, or Amazon Web Services
    e. ScienceSoft Cloud SQL, and so forth.

1. **Object-Oriented Databases:** The kind of database that stores data in the database system using an object-based data paradigm. Similar to how data is saved and displayed in object-oriented programming languages, the data is also saved and displayed as objects.
2. **Hierarchical Databases:** This kind of database keeps information in the form of nodes that show parent-child relationships. Here, the data is organized in a tree-like fashion. Records that have links between them include data. There will only be one parent record for every child record in the tree. However, there may be several child records associated with one parent record.
3. **Network Databases:** The network data model is often followed by the database. Data is shown as a network of connected nodes in this instance. It allows each record to have multiple child and parent nodes to construct a generalized graph structure, unlike a hierarchical database.

Examples of Databases:

Database examples include:

1. **Microsoft SQL Server:** SQL Server is a relational database management system developed by Microsoft. SQL is used in its construction, as it is a recognized query language for DMSs.
2. **Oracle Database:** Oracle Corporation's Oracle Database is based on a multi-model database management system. It is commonly used for doing online transactions.
3. **MySQL:** Relational Database Management System called MySQL. Structured Query Language (SQL) serves as its foundation. It is utilized in e-commerce platforms, data warehouses, and other locations. As a Database Management System for the Web, it is widely utilized.
4. **IBM Db2:** The Db2 Relational Database Management System was created by IBM. Its purpose is to efficiently analyze, store, and retrieve data.
5. **PostgreSQL:** Open source and free to use, PostgreSQL is a Relational Database Management System. It is widely used for data warehousing.

3.5 Relational Database Management System

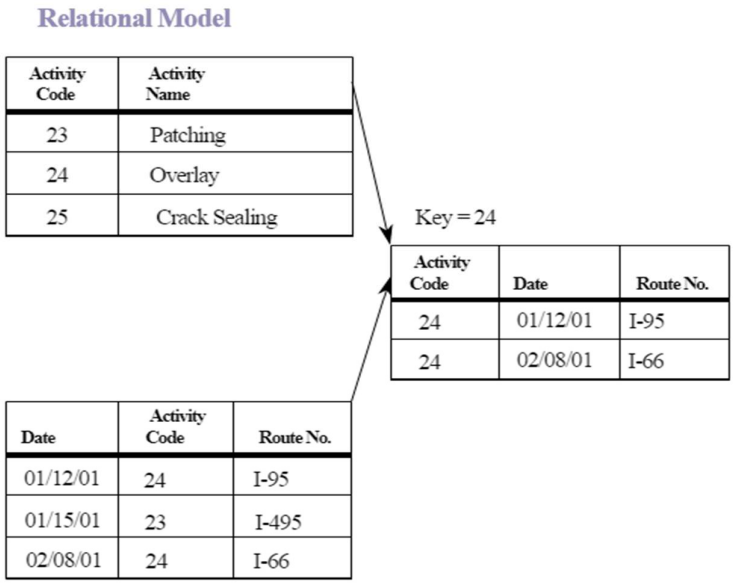
Relational Database Management System is referred to as RDBMS. An application called an RDBMS is used to manage relational databases. All contemporary database systems, including MySQL, Microsoft SQL Server, Oracle, and Microsoft Access, are built on top of RDBMS.

A table is made up of rows and columns and is a grouping of linked data elements.

Each record in the table has specific information stored in a column. Every single entry in a table is called a record, sometimes known as a row. Examine a few samples from the "Customers" table in Northwind:

Brief History of RDBMS

From 1970 to 1972, E.F. Codd published a paper to propose using a relational database model. RDBMS is originally based on E.F. Codd's relational model invention.



(Image Source – Wikipedia)

Table/Relation:

Relationships are the unit of storage for all data in relational databases. Tables are used by the RDBMS database to hold data. A table is made up of rows and columns for storing data and is an arrangement of connected data elements. Every table depicts a real-world entity—a person, location, or event—about which data is gathered. The logical view of the database is the structured grouping of data into a relational table.

Properties of a Relation:

- Each relation has a unique name by which it is identified in the database.
- Relation does not contain duplicate tuples.
- The tuples of a relation have no specific order.
- All attributes in a relation are atomic, i.e., each cell of a relation contains exactly one value.

A table is the simplest example of data stored in RDBMS.

ID	Name	Age	Course
1	Jeet	22	BCA
2	Ravi	21	BBA
3	Rajan	23	B. Tech
4	Suraj	21	BCA
5	Rani	24	MCA

Row or Record:

A record or tuple can also refer to a row in a table. It includes detailed information on every item in the table. In the table, it is a horizontal entity. For instance, there are 5 records in the table above.

ID	Name	Age	Course
1	Jeet	22	BCA

Properties of a Row:

- No two tuples are identical to each other in all their entries.
- All tuples of the relation have the same format and the same number of entries.
- The order of the tuple is irrelevant. They are identified by their content, not by their position.

Column or Attribute:

A column is a vertical element in a table that holds all of the data related to a certain field. For instance, the above table's "name" column has all of the details regarding a student's name.

Name
Jeet
Ravi
Rajan
Suraj

Properties of an Attribute:

- Each property of a relation needs to be given a name.
- The properties allow for null values.
- If no additional value is supplied for an attribute, default values might be specified for the attribute and automatically added.
- The main key is a set of attributes that uniquely identifies each tuple in a relation.

Degree: The total number of attributes that comprise a relation is known as the degree of the table.

For example, the student table has 4 attributes, and its degree is 4.

ID	Name	Age	Course
1	Jeet	22	BCA
2	Ravi	21	BBA
3	Rajan	23	B. Tech
4	Suraj	21	BCA
5	Rani	24	MCA

Cardinality: The cardinality of a table is the total number of tuples in a relation at any one time. An empty table is a relation with cardinality zero.

The student table, for instance, has five rows means five cardinality.

ID	Name	Age	Course
1	Jeet	22	BCA
2	Ravi	21	BBA
3	Rajan	23	B. Tech
4	Suraj	21	BCA
5	Rani	24	MCA

- **Domain:** Each attribute's potential values are indicated by its domain. Standard data types like integers, floating numbers, etc., can be used to specify it. For instance, the values for the attribute Marital Status may only have married or single values.
- **NULL Values:** The table's NULL value indicates that the field was left empty when the record was created. It is not the same as a field with spaces in it or a value that is filled with zeros.
- **Data Accuracy:** With every RDBMS, the following categories of data integrity are present:
- **Entity integrity:** It stipulates that a table's rows cannot be duplicates.
- **Domain integrity:** By limiting the type, format, or range of values, it ensures that entries for a certain column are valid.
- **Referential integrity:** It states that rows that are used by other records cannot be removed.
- **User-defined integrity:** It upholds certain, user-defined business standards. These regulations differ from those pertaining to referential, entity, and domain integrity.

3.6 Database Design

Database design is a group of procedures that makes enterprise data management system design, development, deployment, and maintenance easier. A well-designed database reduces maintenance costs,

enhances data consistency, and uses less disk space. The database designer determines which data needs to be stored and how the data items relate to one another.

Creating logical and physical design models of the suggested database system is the primary goal of database design in database management systems (DBMS).

The logical model ignores physical considerations and focuses on the data requirements and data to be stored. The method and location of the data's physical storage are unimportant to it.

Using hardware resources and software technologies like database management systems (DBMS), the physical data design model translates the database's logical DB design into physical media.

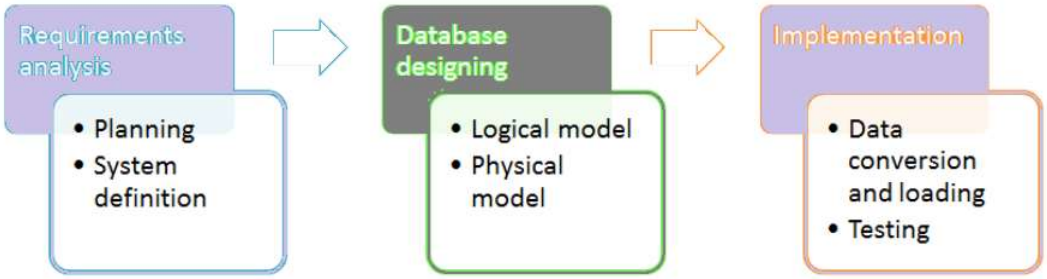
Why Database Design is Important?

It helps produce database systems

1. That meet the requirements of the users
2. Have high performance.

Database design process in DBMS is crucial for **high performance** database system.

Note, the genius of a database is in its design. Data operations using SQL is relatively simple



When designing database systems, there are several steps in the database development life cycle that must be followed. It is not necessary to adhere to the development life cycle's steps in a strict chronological order.

Database design is typically a fairly straightforward procedure with few phases on small database systems.

To have a complete understanding of the above graphic, let's examine the specific elements mentioned in each phase for a summary of the DBMS design process.

Requirements Analysis

- **Planning:** The planning of the full Database Development Life Cycle is the focus of this stage of database design concepts. It considers the organization's information systems strategy.
- **System Definition:** In this phase, the parameters and extent of the suggested database system are established.

Database Designing

- **Logical Model:** Creating a database model according to specifications is the focus of this step. Without any physical implementations or particular DBMS considerations, the entire design is written down.
- **Physical Model:** Taking into consideration the DBMS and physical implementation elements, this stage implements the database's logical model.

Implementation

- **Data Conversion and Loading:** this stage of relational databases design is concerned with importing and converting data from the old system into the new database.
- **Testing:** this stage is concerned with the identification of errors in the newly implemented system. It checks the database against requirement specifications.

3.7 Database Normalization

A database design method called normalization helps to minimize redundant data and get rid of unwanted features like Insertion, Update, and Deletion Anomalies. Larger tables are divided into smaller tables by normalization procedures, which then use relationships to connect them. In SQL, normalization serves to guarantee that data is stored logically and to remove redundant or repetitive data.

Edgar Codd, the creator of the relational model, introduced the First Normal Form, which introduced the notion of data normalization. He then went on to expand the theory with the Second and Third Normal Forms. Afterwards, he collaborated with Raymond F. Boyce to advance the Boyce-Codd Normal Form theory.

Types of Normal Forms in DBMS

Here is a list of Normal Forms in SQL:

First Normal Form, or 1NF: It makes assurance that every record in the database table is distinct and that every column has atomic (indivisible) values. By doing this, repetitive groupings are removed, allowing data to be organized into tables and columns.

Second Normal Form, or 2NF: The redundant data in a table that is applied to several rows needs to be taken out and put into different tables. Every non-key attribute needs to work flawlessly on the main key.

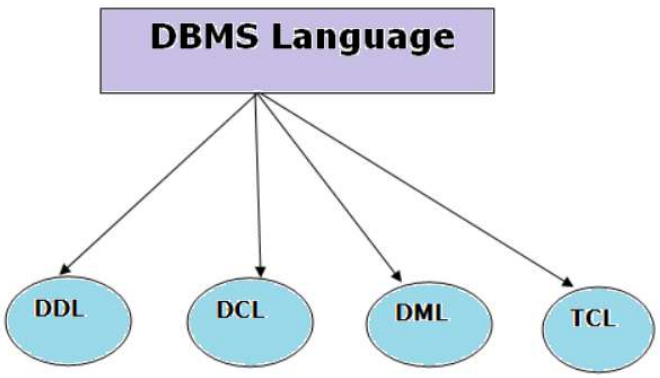
The Third Normal Form, or 3NF: It expands on the Second Normal Form by guaranteeing that every non-key feature is independent of the others and fully functional on the primary key. Thus, transitive reliance is removed.

Boyce-Codd Normal Form, or BCNF: It is an improvement on 3NF that takes care of abnormalities that 3NF is unable to handle. It guarantees even tighter adherence to normalization criteria by making each determinant a candidate key.

The Fourth Normal Form, or 4NF: It takes multivalued dependencies into account. It guarantees that no record contains more than one independent set of multivalued facts about the same entity.

3.8 Database Languages

Types of Database Languages:



1. Data Definition Language (DDL)

- **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under **DDL**:

- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

2. Data Manipulation Language (DML)

DML stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under **DML**:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

3. Data Control Language (DCL)

- **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.

(But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under **DCL**:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

4. Transaction Control Language (TCL)

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under **TCL**:

- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit.

3.9 Question and Answers

1. **What is data, and why is it important for organizations?**

   Answer: Data refers to raw facts and figures that are collected, processed, and used for various purposes. It is important for organizations because it provides the basis for making informed decisions, understanding trends, improving processes, and gaining competitive advantages. By analyzing data, organizations can derive valuable insights that help in strategic planning, operational efficiency, and customer satisfaction.

2. **Explain the difference between structured, unstructured, and semi-structured data with examples.**

   Answer:

- **Structured Data**: This is highly organized data that is easily searchable in databases. It is typically stored in rows and columns. Example: A table in a SQL database containing customer information such as name, address, and phone number.
- **Unstructured Data**: This type of data does not have a predefined format or organization, making it more challenging to analyze. Example: Text documents, emails, and social media posts.
- **Semi-Structured Data**: This data does not conform to a rigid structure but contains tags or markers to separate elements. Example: JSON and XML files.

3. **What are the main advantages of using a DBMS?**

   Answer: The main advantages of using a DBMS include:

- **Data Redundancy and Inconsistency Reduction**: Ensures that data is integrated and centrally managed, reducing duplication.
- **Data Integrity**: Enforces constraints to maintain accurate and consistent data.
- **Data Security**: Provides access controls to ensure only authorized users can access or modify data.
- **Efficient Data Access**: Uses sophisticated algorithms to store and retrieve data efficiently.
- **Concurrent Access and Crash Recovery**: Supports multiple users accessing data simultaneously and can recover from crashes.
- **Backup and Recovery**: Provides automated procedures to protect data against loss or corruption.

4. **Describe the key types of databases and their specific use cases.**

   Answer:

- **Relational Databases (RDBMS)**: Use structured query language (SQL) and are ideal for managing structured data with relationships. Example: Customer databases, financial systems.
- **NoSQL Databases**: Designed for unstructured or semi-structured data, providing flexibility and scalability. Example: Social media platforms, real-time analytics.
- **Object-Oriented Databases**: Store data in objects, similar to object-oriented programming. Example: Complex data modelling applications.
- **Distributed Databases**: Spread across multiple locations or devices, providing high availability and fault tolerance. Example: Cloud storage solutions.

5. **What is database normalization, and why is it important?**

   Answer: Database normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing a database into tables and defining relationships between them according to rules designed to protect the data and make the database more flexible by eliminating redundancy and inconsistent dependency. Normalization is important because it ensures efficient data storage, reduces redundancy, improves data integrity, and makes it easier to maintain and update the database.

3.10 References

- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.
- Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.
- Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*, 13(6), 377-387.
- Stonebraker, M., & Hellerstein, J. M. (2001). "Content Management in Relational Databases: The Next Generation." *Communications of the ACM*, 44(3), 111-118.
- ISO/IEC 9075:2016: Information technology – Database languages – SQL.
- W3C. (n.d.). "XML and Databases." World Wide Web Consortium (W3C). Retrieved from https://www.w3.org/XML/

***