CONCEPTS OF DATABASE MANAGEMENT SYSTEMS : CHAPTER 09

1. **What is a database?**

   A **database** is an organized collection of structured information or data that can be easily accessed, managed, and updated. Databases store information in tables, making it easy to retrieve, manipulate, and analyze data efficiently. They are essential for storing large amounts of data in a structured format and are widely used across many applications, from small personal projects to large enterprise systems.

2. **Distinguish between DBMS and RDBMS.**

   A **DBMS** (Database Management System) manages data in files or simple tables, generally without strict relationships between data. It's suitable for smaller, simpler applications, as it lacks advanced integrity constraints and data normalization, which can lead to redundancy. Examples of DBMS include Microsoft Access, dBASE etc.

   A **RDBMS** (Relational Database Management System), on the other hand, organizes data into related tables with defined relationships (using primary and foreign keys), ensuring data accuracy and consistency. RDBMS supports normalization, follows ACID (Atomicity, Consistency, Isolation, Durability) properties for reliable transactions, and includes robust security and scalability features, making it ideal for complex and large-scale applications. Examples of RDBMS are MySQL, MS SQL Server, PostgreSQL and Oracle.

3. **What are tuples and attributes?**

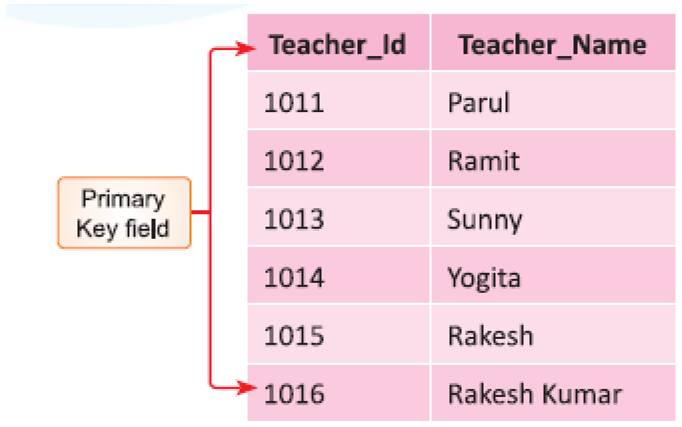
   Tuples are the individual rows in a table. Each tuple represents a single record, or data item, in the table, containing data across multiple fields (columns) for a specific entity or item. For example, in a **Customers** table, a single row representing one customer (with their ID, name, and contact information) is a tuple.

   Attributes are the columns in a table. Each attribute defines a specific property or characteristic of the data stored in the table, representing one type of information for each record. For example, in the **Customers** table, attributes could include `CustomerID`, `FirstName`, `LastName`, and `Email`, where each attribute holds a particular type of information about each customer.

4. **Describe primary key.**

   A **primary key** is a unique identifier for each record in a table, ensuring that each row is distinct and contains no duplicate or `NULL` values. It's essential for maintaining data integrity within the table, as it uniquely identifies each entry. It is immutable i.e once created cannot be changed.

   Example: Teacher_ID in Teacher_Data Table.



Primary Key field

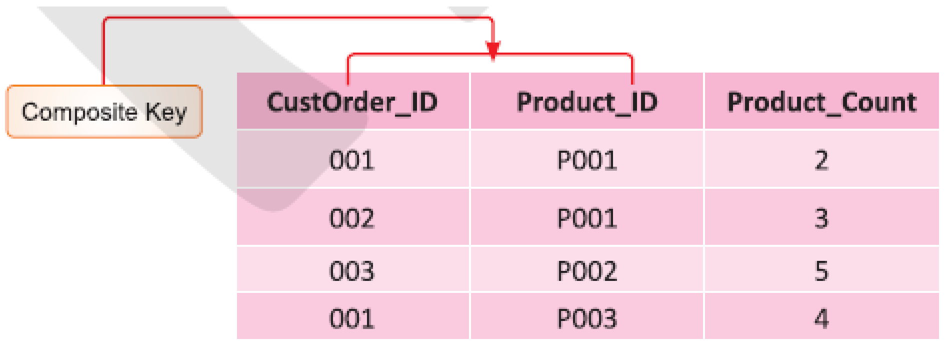
Teacher_Id	Teacher_Name
1011	Parul
1012	Ramit
1013	Sunny
1014	Yogita
1015	Rakesh
1016	Rakesh Kumar

5. **What is a composite key? Give examples.**

   A **composite key** is a combination of two or more columns in a table that together uniquely identify each record in the table. Composite keys are used when a single column is not enough to create a unique identifier for each record, so multiple columns are combined to ensure uniqueness. Each component of a composite key individually may not be unique, but together they provide a unique identifier for the records.

   Example of a Composite Key

   Consider the following sales table. In the sales table CustOrder_ID and Product_ID are both unique from which the records can be retrieved uniquely. Such combinations form composite keys.


Composite Key

CustOrder_ID	Product_ID	Product_Count
001	P001	2
002	P001	3
003	P002	5
001	P003	4

6. **Give one difference between network and hierarchical data models.**

   The **Hierarchical Data Model** organizes data in a **tree-like structure** with a strict **one-to-many relationship**, where each child has only one parent. It follows a **top-down approach** for navigation, making it less flexible.

   In contrast, the **Network Data Model** arranges data in a **graph-like structure** that supports **many-to-many relationships**, allowing a child to have multiple parents.

CREATING AND USING TABLES (DBMS) : CHAPTER 10

---

1. **Describe the different database objects.**

   In a database, objects are the essential elements that store, organize, and manage data. The main types of database objects include **Tables**, **Queries**, **Forms**, **Reports**, **Macros**, and **Modules**.

   1. Tables: The core objects in a database, tables store data in rows and columns.Each table consists of columns (also known as fields) representing attributes of the data and rows (also known as records) representing individual data entries.

   2. Queries: Queries retrieve specific data from one or more tables based on specified criteria. You can use queries to filter, sort, and analyze data or to combine data from multiple tables.

   Common query types include **Select Queries** (for viewing data), **Update Queries** (for modifying data), **Delete Queries** (for removing data), and

Append Queries (for adding new records).

3. Forms: Forms provide a user-friendly interface for data entry, modification, and navigation within the database.Forms often display fields from tables or queries and can include controls like text boxes, buttons, and drop-down lists.

4. Reports: Reports are formatted outputs used for summarizing and presenting data in a readable and printable format. Reports can display data from tables or queries and often include grouping, sorting, and aggregation to provide insights.

5. Macros: Macros automate repetitive tasks in the database, such as opening forms, running queries, or validating data. Macros consist of predefined actions or steps that are executed in sequence.

6. Modules: Modules contain custom-written code, typically in a programming language like VBA (Visual Basic for Applications), to extend database functionality. Modules consist of functions and procedures that can perform complex calculations or data processing.

2. **What are data types?**

Data types are classifications that define the kind of data a field in a database can hold. In databases and programming, data types are essential because they determine how data is stored, processed, and validated. Following are common data types and their purposes:

1. Text/String: Stores text or alphanumeric characters.String/Text are of two types Char (fixed length), Varchar (variable length).

Examples: Names, addresses, descriptions.

2. Number: Stores numeric data used for calculations. Number can vary like Integer (whole numbers), Float/Real (decimal numbers), Double (higher precision decimals).

Examples: Age, quantity, salary.

3. Date/Time: Stores dates, times, or combinations of both. Variants like

Date, Time, DateTime, Timestamp are available.

Examples: Birthdate, event time, timestamp.

4. Boolean: Stores logical values (true or false).

Examples: Yes/No, True/False, 1/0 (often used for binary choices or status indicators).

5. Currency: Stores monetary values with precision. Often includes specific formatting and precision for decimal places.

Examples: Price, revenue, salary.

6. AutoNumber/Serial: Stores unique sequential numbers automatically generated by the database. Often used as primary keys to uniquely identify records.

Examples: ID numbers, order numbers.

7. Binary: Stores binary data (non-text files). Blob (Binary Large Object) for storing large files.

Examples: Images, audio files, documents.

8. Large Object Data Types: Stores large text or binary data. Varies like Blob (Binary), Clob (Character Large Object).

Examples: Long text fields, large files like images or PDFs.

9. Enumerated/Enum: Stores a predefined list of values.Ensures that a field contains only specific allowed values.

Examples: Status values like "Pending," "In Progress," "Completed."

3. **Explain the user of Text[VARCHAR] datatype.**

The **Text** or **VARCHAR** (Variable Character) data type in databases is used to store alphanumeric text, such as letters, numbers, symbols, and spaces, with variable lengths. Unlike fixed-length types, VARCHAR only uses the storage space required for each entry, up to a maximum limit defined for the field. This makes it memory-efficient, as it doesn't waste space on unused characters. Common uses for VARCHAR include storing names, addresses, emails, descriptions, and comments—any field where text length varies between entries. For instance, in a customer database, a `FirstName` field might be set to `VARCHAR(50)` to allow names up to 50 characters long, while an `Address` field might be set to `VARCHAR(255)` to accommodate longer entries. This flexibility makes VARCHAR a popular choice for text fields, ensuring efficient storage and performance.

4. **Distinguish between the table data and design views?**

   Table Data View and **Design View** are two modes for working with tables in database software like Microsoft Access or OpenOffice Base. They serve different purposes, focusing on either data entry or table structure.

   1. Data View (or Datasheet View): Allows users to view, enter, and edit data in the table.Displays the table in a grid format, showing rows (records) and columns (fields) where you can directly input and modify data. Ideal for tasks involving data manipulation, such as adding, editing, or deleting records.

   Limitations: Doesn't allow changes to the structure of the table, like adding new fields or setting data types.

   2. Design View: Allows users to define and modify the structure of the table. Displays fields in a list format with options to set field names, data types, and field properties (e.g., default values, field size, validation rules). Ideal for creating new tables or modifying the design of existing tables, such as adding or deleting fields, setting primary keys, and defining relationships.

   Limitations: Doesn't allow data entry or editing of individual records, focusing only on table design.

5. **How do you edit a table structure?**

   To edit a table's structure, you need to access the Design **View** in your database application. Design View allows you to make changes to field names, data types, and other properties of the table without affecting the data itself.

Following are the steps to do

1. Open design view
2. Modify filed names, required
3. Change Data Types, required
4. Add / Delete required fields
5. Set / Change the Field properties
6. Save the changes (Ctrl + S)

PERFORMING OPERATIONS ON TABLES : CHAPTER 11

---

1. **What is the use of tables in a database?**

   Tables are the primary structures in a database used to store and organize data in a structured, accessible way. They consist of **rows** (records) and **columns** (fields), with each row representing an individual record and each column representing a specific attribute of that record. Following are the uses of tables in a database.

   a. **Data Storage**: Tables serve as containers for data in the database, where each table represents a different entity or subject.

      For example, a **Customers** table stores customer data, while an **Orders** table stores order information.

   b. **Data Organization**:By structuring data into rows and columns, tables keep information organized, making it easy to manage, sort, and retrieve data. Columns define what data can be stored (e.g., names, dates, amounts), and each row holds a unique record.

   c. **Data Relationships**: Tables in a relational database can be linked to each other through **primary keys** (unique identifiers for records in a table) and **foreign keys** (references to primary keys in other tables). These relationships help organize related data across multiple tables.

      For example, in a sales database, a **CustomerID** in the Customers table might link to a **CustomerID** in the Orders table, creating a relationship between the tables.

d. **Efficient Data Retrieval and Analysis**: Tables support database operations like queries, sorting, filtering, and reporting, which allow for efficient data retrieval. For example, a query can quickly find all orders made by a specific customer.

e. **Data Integrity and Validation**: Tables allow for data validation through constraints, ensuring that data is accurate and consistent. Field properties like data type, length, and required fields can enforce standards across data entries.

f. For example, a **Date** field can enforce valid date entries only, or a **Not Null** constraint ensures that certain fields aren't left blank.

g. **Data Scalability**: Tables enable databases to handle vast amounts of data by storing it in manageable, structured blocks. As databases grow, tables can store increasingly complex datasets without compromising data accessibility or structure.

2. **How can you sort records of a table in descending order?**

Records of a table are sorted as followed

Open the Table or Query: Open the table directly or use a query to work with the data.

Select the Field to Sort: Go to the column you want to sort by.

Apply Descending Sort:

a. Right-click on the field name (column header) and select **Sort Descending**.
b. Or, in the **Sort & Filter** group on the toolbar, click **Descending** (usually represented by a "Z-A" or "↓" icon).

One can use SQL as

```
SELECT * FROM TableName ORDER BY FieldName DESC;
```

3. **Describe referential integrity.**

Referential integrity is a concept in databases that ensures the accuracy and consistency of data between related tables. It uses **primary keys**

(unique identifiers) and **foreign keys** (links to primary keys in other tables) to create relationships between tables. Referential integrity ensures that a foreign key in one table always points to a valid record in another table, preventing orphaned records (records that reference non-existent data). It also enforces rules like **cascade updates** (updating related foreign keys when a primary key is changed) and **cascade deletes** (automatically deleting related records when a primary record is deleted). This helps maintain data consistency and avoids errors across tables.

4. **List different types of relationships.**

In relational databases, relationships define how tables are connected to each other. There are three primary types of relationships:

1. One-to-One (1:1) Relationship: In a one-to-one relationship, each record in Table A is related to exactly one record in Table B, and vice versa. This type of relationship is used when information in two tables is closely related and could logically be stored in a single table but is split for design or security reasons.

Example: A **Person** table and a **Passport** table, where each person can have only one passport, and each passport is assigned to only one person.

2. One-to-Many (1) Relationship: In a one-to-many relationship, each record in Table A is related to multiple records in Table B, but each record in Table B is related to only one record in Table A. This is the most common type of relationship and is used when one record in a table can be associated with many records in another table.

Example: A **Customer** table and an **Orders** table, where each customer can place multiple orders, but each order is placed by only one customer.

3. Many-to-Many (M) Relationship: In a many-to-many relationship, multiple records in Table A are related to multiple records in Table B. Many-to-many relationships are implemented by creating a **junction table** (also called a **linking** or **associative table**) that holds the foreign keys from both tables, which eliminates the need for direct many-to-many connections between the tables.

Example: A **Students** table and a **Courses** table, where each student can enroll in multiple courses, and each course can have multiple students.

5. **Give an example of a many-to-many relationship.**

A classic example of a **many-to-many (M) relationship** is the relationship between **Students** and **Courses** in a school or university database.

Example: Students and Courses

- **Students Table**: Contains information about students, such as `StudentID`, `FirstName`, `LastName`, etc.
- **Courses Table**: Contains details about courses, such as `CourseID`, `CourseName`, `Instructor`, etc.

Problem:

- Each student can enroll in multiple courses (e.g., a student might take Math, History, and Science).
- Each course can have multiple students enrolled (e.g., Math may have 30 students).

This creates a **many-to-many** relationship:

- One student can be enrolled in many courses.
- One course can have many students.

Solution: Junction Table (Enrollment Table)

To handle this many-to-many relationship, we introduce a third table, called a **junction table** (or **linking table**), which links the two tables. This table contains foreign keys from both the **Students** and **Courses** tables.

Enrollment Table:

- `StudentID` (foreign key referencing `Students` table)
- `CourseID` (foreign key referencing `Courses` table)

This table doesn't store additional data, but instead creates a relationship between students and courses.

6. **Give two advantages of relating a table in a database?**

   Relating tables in a database provides several advantages, yet following are two key benefits:

   1. **Eliminates Data Redundancy**
      By linking related tables, you avoid storing duplicate data. For example, instead of storing customer details in every order record, you can create a separate **Customers** table and relate it to an **Orders** table. This saves storage space and improves data integrity.

   2. **Ensures Data Consistency and Accuracy**
      Relationships enforce **referential integrity**, meaning that data in related tables stays accurate and consistent. For example, if an **Orders** table references a **Customers** table, you cannot add an order for a non-existent customer. This prevents orphaned records and maintains logical data connections.

RETRIEVING DATA USING QUERIES : CHAPTER 12

---

1. **What is a query?**

   A query is a database object that enables one to retrieve records from one or more tables of the database or different databases that meet a specific condition or criteria. Query also performs operations like insert, update or delete records. It also allows users to join and filter data or records.

2. **List the different Query Views of Open office Base.**

   OpenOffice Base provides the following **Query Views** to help users create and manage queries in a database:

   - **Design View**:
     This is a graphical interface where users can build queries by selecting tables, fields, and specifying criteria without needing to write SQL code. It is ideal for beginners or those unfamiliar with

SQL.

- **SQL View**:
  This view allows advanced users to directly write and execute **SQL (Structured Query Language)** commands. It provides flexibility for creating complex queries, including those not supported by the graphical interface.
- **Query Wizard**:
  A step-by-step guide that helps users create queries without any prior knowledge of SQL or database structure. The wizard asks a series of questions to define the fields, criteria, and sorting preferences for the query.

3. **What is the difference between Query Wizard and Query design View?**

The **Query Wizard** and **Query Design View** in OpenOffice Base are two methods for creating and editing queries, but they differ in their approach, flexibility, and complexity:

Query Wizard

Designed for beginners or users who prefer a guided process for creating queries. Provides a step-by-step wizard that walks you through selecting tables, fields, criteria, sorting, and other options. Very user-friendly, with no need for SQL knowledge or technical expertise. Limited in complexity; suitable for simple queries but may not support advanced features like calculated fields or custom joins.

Query Design View

Allows users to create and modify queries directly in a graphical interface, offering more control and flexibility. Displays tables, fields, and relationships graphically, and lets you define criteria, sorting, and groupings manually. Requires a basic understanding of databases and query logic but doesn't demand direct SQL knowledge. More flexible than the wizard; supports complex queries, calculated fields, and multiple table joins.

4. **What is SQL?**

   SQL (Structured Query Language) is a standardized programming language used to manage and manipulate data in relational databases. It provides a structured way to interact with databases, enabling users to perform operations like retrieving, inserting, updating, deleting, and defining data.

5. **Distinguish between the DLL and DML commands.**

   DDL (Data Definition Language) commands are used to define, modify, and delete the structure of database objects such as tables, indexes, and databases. These commands operate on the database rather than the data and include commands like `CREATE`, `ALTER`, `DROP`, and `TRUNCATE`. For example, `CREATE TABLE Students (ID INT, Name VARCHAR(50));` defines a new table. DDL changes are usually auto-committed, meaning they are permanent and cannot be rolled back.

   DML (Data Manipulation Language) commands, on the other hand, are used to manipulate the actual data stored in the database tables. These include commands like `SELECT`, `INSERT`, `UPDATE`, and `DELETE`. For example, `INSERT INTO Students (ID, Name) VALUES (1, 'John');` adds a record to the table. DML changes can be rolled back or committed using transaction control commands like `COMMIT` and `ROLLBACK`.

6. **Name the ways of creating a query in Libre Office Base and Explain them.**

   In **LibreOffice Base**, queries are used to retrieve specific data from a database based on certain conditions. There are three primary ways to create a query:

   1. Using the Design View

   This is a visual method where users can create queries without writing SQL code.

- Open **LibreOffice Base** and go to the **Queries** section.
- Click on **"Create Query in Design View"**.
- Add the tables or queries from which you want to retrieve data.
- Select the required fields, apply sorting, and set criteria for filtering data.
- Save and run the query to view results.

2. Using the SQL View (Direct SQL Mode)

This method allows users to write SQL (Structured Query Language) statements directly.

- Open **LibreOffice Base** and navigate to the **Queries** section.
- Click on **"Create Query in SQL View"**.
- Type your SQL query (e.g., `SELECT * FROM Customers WHERE City = 'New York';`).
- Save and execute the query to retrieve results.

3. Using the Query Wizard

The Query Wizard guides users step-by-step to create a query with minimal effort.

- Open **LibreOffice Base** and go to the **Queries** section.
- Click on **"Use Wizard to Create Query"**.
- Select the table and fields you want to include.
- Set sorting preferences, define search conditions, and choose grouping options.
- Preview the query, give it a name, and save it.
