Slide 1: Title Slide

- **Title:** Database Management Systems
- **Subtitle:** Exploring SQL, Data Manipulation, and Advanced Concepts
- **Presenter:** [Your Name]
- **Date:** [Date]

Slide 2: Introduction to SQL Database Creation

- **Title:** SQL Database Creation
- **Explanation:**
  - SQL (Structured Query Language) is the standard language for managing relational databases.
  - Creating a database is the first step in organizing and storing data.
- **Syntax:**
  - CREATE DATABASE database_name;
- **Example:**
  - CREATE DATABASE my_first_database;
- **Image:**

Slide 3: DDL Commands

- **Title:** Data Definition Language (DDL)
- **Explanation:**
  - DDL commands are used to define the structure of the database.
  - They deal with creating, modifying, or deleting database objects.
- **Commands:**
  - CREATE: Create database objects (tables, views, etc.)
    - CREATE TABLE table_name (column1 datatype, column2 datatype, ...);
  - ALTER: Modify the structure of an existing object
    - ALTER TABLE table_name ADD column_name datatype;
  - DROP: Delete database objects
    - DROP TABLE table_name;
- **Example:**

  -
    ```
    CREATE TABLE Employees (
        EmployeeID int,
        FirstName varchar(255),
        LastName varchar(255)
    );
    ```
- **Image:** [Image illustrating DDL commands and their effects on database schema]

Slide 4: DML Commands

- **Title:** Data Manipulation Language (DML)
- **Explanation:**
  - DML commands are used to manipulate the data within database objects.
  - They deal with inserting, retrieving, updating, and deleting data.
- **Commands:**
  - SELECT: Retrieve data from the database
    - SELECT column1, column2 FROM table_name WHERE condition;
  - INSERT: Insert new data into a table
    - INSERT INTO table_name (column1, column2) VALUES (value1, value2);
  - UPDATE: Modify existing data in a table
    - UPDATE table_name SET column1 = value1 WHERE condition;
  - DELETE: Delete data from a table
    - DELETE FROM table_name WHERE condition;
- **Example:**

  -
    ```
    INSERT INTO Employees (EmployeeID, FirstName, LastName) VALUES (1, 'John', 'Doe');
    SELECT * FROM Employees;
    UPDATE Employees SET FirstName = 'Jane' WHERE EmployeeID = 1;
    DELETE FROM Employees WHERE EmployeeID = 1;
    ```

- **Image:** [Image illustrating DML operations on a database table]

Slide 5: Advanced Data Handling

- **Title:** Advanced Data
- Explanation:
  * Beyond basic data types, databases support complex structures.
- **Examples:**
  - BLOB (Binary Large Object): Stores binary data (images, audio, video).
  - CLOB (Character Large Object): Stores large text data.
  - JSON: Stores JSON documents.
  - XML: Stores XML documents.
  - Spatial Data: Stores geographic information.
- **Example:**

  -
    ```
    CREATE TABLE Documents (
        DocumentID int,
        FileName varchar(255),
        FileData BLOB
    );
    ```

- **Image:** [Image showing different types of advanced data being stored in a database]

Slide 6: Cursors

- **Title:** Cursors
- **Explanation:**
  - Cursors are database objects that allow you to access and manipulate data row by row.
  - They are used in procedural database code (e.g., stored procedures).
- Life Cycle:
  * Declare: Define the cursor.
  * Open: Open the cursor to make it ready for fetching.
  * Fetch: Retrieve data from the result set, row by row.
  * Close: Close the cursor to release resources.
  * Deallocate: Remove the cursor definition.
- Fetch Commands
  * FETCH NEXT: Retrieves the next row in the result set.
- Advantages:
  * Row-by-row processing
  * Complex data manipulation
- **Image:** [Diagram illustrating the cursor life cycle]

Slide 7: Triggers

- **Title:** Triggers
- **Explanation:**
  - Triggers are special stored procedures that automatically execute in response to certain events in a database.
- **Types:**
  - BEFORE: Trigger fires before the event.
  - AFTER: Trigger fires after the event.
  - INSTEAD OF: Trigger replaces the event (for views).
- **Events:**
  - INSERT
  - UPDATE
  - DELETE
- **Example:**

  -
    ```
    CREATE TRIGGER update_timestamp
    BEFORE UPDATE ON Employees
    FOR EACH ROW
    ```

```
  SET NEW.LastModified = NOW();
```

- **Image:** [Flowchart showing how triggers work in response to database events]

Slide 8: Procedures and Functions

- **Title:** Procedures and Functions
- **Procedures:**
    - Stored programs that perform specific tasks.
    - Can have input and output parameters.
    - Do not return a value.
    - Example:
      ```
      CREATE PROCEDURE GetEmployeeName(IN EmpID INT, OUT EmpName
      VARCHAR(255))
      BEGIN
        SELECT FirstName FROM Employees WHERE EmployeeID = EmpID INTO
      EmpName;
      END;
      ```
- **Functions:**
    - Stored programs that perform calculations and return a single value.
    - Can have input parameters.
    - Must return a value.
    - Example:
      ```
      CREATE FUNCTION CalculateSalary(Salary INT, Bonus INT)
      RETURNS INT
      BEGIN
        DECLARE TotalSalary INT;
        SET TotalSalary = Salary + Bonus;
        RETURN TotalSalary;
      END;
      ```
- **Image:** [Code snippets illustrating a procedure and a function]

Slide 9: Indexing

- **Title:** Indexing
- **Explanation:**
    - Indexing is a database optimization technique used to speed up data retrieval.
    - An index is a data structure that provides a quick way to locate specific rows in a table.

- Types:
  * Clustered index
  * Non-clustered index
- **Example:**
  - CREATE INDEX idx_lastname ON Employees (LastName);
- **Image:** [Diagram illustrating how an index works to speed up data retrieval]

Slide 10: Access Control (DCL)

- **Title:** Access Control - Data Control Language (DCL)
- **Explanation:**
  - DCL commands are used to control access to database objects.
  - They deal with granting and revoking privileges.
- **Commands:**
  - GRANT: Assign privileges to users or roles.
    - GRANT SELECT, INSERT ON table_name TO user_name;
  - REVOKE: Take back privileges from users or roles.
    - REVOKE SELECT ON table_name FROM user_name;
- **Image:** [Diagram showing how DCL commands control access to database resources]

Slide 11: Database Security

- **Title:** Database Security
- **Explanation:**
  - Database security refers to the measures taken to protect a database from unauthorized access, modification, or destruction.
- **Need for Database Security:**
  - Prevent data breaches
  - Maintain data integrity
  - Ensure compliance with regulations
  - Protect sensitive information
- **Examples:**
  - Strong passwords
  - Access control lists
  - Encryption
  - Firewalls
  - Regular security audits
- **Image:** [Image representing database security]

Slide 12: De-normalization

- **Title:** De-normalization

- **Explanation:**
  - De-normalization is a database optimization technique where redundant data is added to a table to improve read performance.
  - It is the opposite of normalization.
- **Example:**
  - Consider an Orders table and a Customers table. In a normalized database, they would be separate. In a denormalized database, you might add customer name to the Orders table to avoid a join.
- **Image:** [Diagram illustrating normalization vs. de-normalization]

Slide 13: Database Tuning

- **Title:** Database Tuning
- **Explanation:**
  - Database tuning is the process of optimizing database performance.
  - It involves adjusting various parameters and settings to improve speed and efficiency.
- **Techniques:**
  - Indexing
  - Query optimization
  - Caching
  - Partitioning
  - Hardware upgrades
- **Image:** [Dashboard showing database performance metrics]

Slide 14: Clustering

- **Title:** Clustering
- **Explanation:**
  - Clustering involves storing related data together on disk to improve I/O performance.
- **Architectures**:
  - Shared-disk: All nodes can access all data.
  - Shared-nothing: Each node has its own data.
- **Image:** [Diagram illustrating shared-disk and shared-nothing clustering architectures]

Slide 15: Conclusion

- **Title:** Conclusion
- **Summary:**
  - SQL is a powerful language for database management.
  - DDL and DML commands are essential for defining and manipulating data.

    - Advanced concepts like cursors, triggers, and stored procedures enhance database functionality.
    - Database security, indexing, de-normalization, tuning, and clustering are crucial for optimizing performance and ensuring data integrity.
- **Thank you!**
- **Questions?**