7 Creating and Populating a Database

Creating and Populating a Database

Main Points:

Creating Tables and Relationships

Defining Fields and Attributes

Importing Data from External Sources

Populating Tables with Initial Data

Creating Tables and Relationships

Creating Tables:

- Tables store data in rows and columns.
- **Example:** `Customers`, `Orders`, `Products`.

Defining Relationships:

- Primary and Foreign Keys link tables.
- **One-to-Many Relationship Example:**
    - A **Customer** can place many **Orders**, but each order is linked to only one customer.

Defining Fields and Attributes

What are Fields/Attributes?

- Fields are the columns in a table; attributes describe entity properties.

Key Points:

- **Data Types:** Ensure correct data types (e.g., `INT`, `VARCHAR`, `DATE`).
- **Constraints:** Enforce rules like `NOT NULL`, `UNIQUE`, `DEFAULT`.

Examples:

- **Customer Table:**
  - Fields: `CustomerID`, `Name`, `Email`
  - Data Types: `INT`, `VARCHAR(100)`, `VARCHAR(100)`

Importing Data from External Sources

Why Import Data?

- Speed up database population by importing large datasets.

Common Sources:

- **Spreadsheets (CSV, Excel)**
- **Text Files (TXT)**

Tools for Import:

- SQL Server Import and Export Wizard.
- Bulk Insert commands (`BULK INSERT`, `OPENROWSET`).

Steps to Import Data

Prepare the Data:

- Ensure data formatting matches the database schema.
- Clean and validate the data before import.

Importing Data:

- Use import tools (e.g., `BULK INSERT`, `SQL Server Management Studio`).
- **Example Command:**

```
BULK INSERT Customers

FROM 'C:/data/customers.csv'

WITH (FIRSTROW = 2, FIELDTERMINATOR = ',', ROWTERMINATOR = '\n');
```

Populating Tables with Initial Data

Manual Entry:

- For small datasets, you can manually insert records.
- **Example:**

```
INSERT INTO Customers (CustomerID, Name, Email)

VALUES (1, 'John Doe', 'john.doe@example.com');
```

Bulk Inserts:

- For larger datasets, importing from external sources or bulk inserting is more efficient.

Best Practices

Data Validation: Ensure data is clean before importing.

Backup: Always backup the database before large data imports.

Testing: Test your data import process on smaller datasets first.

Querying Data from Multiple Tables

Concept: Combining data from multiple tables using **JOIN** operations.

Types of Joins:

- **INNER JOIN:** Returns records that have matching values in both tables.
- **LEFT JOIN:** Returns all records from the left table and matching records from the right table.
- **RIGHT JOIN:** Returns all records from the right table and matching records from the left table.
- **FULL OUTER JOIN:** Returns all records when there is a match in either table.

```
SELECT Customers.Name, Orders.OrderDate, Orders.Amount

FROM Customers

INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

Using Logical Operators (AND, OR, NOT)

Logical Operators in Queries:

- **AND:** Both conditions must be true.
- **OR:** At least one condition must be true.
- **NOT:** Excludes records that meet a specific condition.

Examples:

- **AND:**

```
SELECT * FROM Orders
WHERE Amount > 100 AND OrderDate >= '2023-01-01';
```

OR:

```
SELECT * FROM Orders
WHERE Amount > 100 OR Amount < 50;
```

NOT:

```
SELECT * FROM Customers
WHERE NOT Country = 'USA';
```

Creating Queries with Multiple Criteria

Multiple Criteria Queries: Combine conditions using logical operators and aggregate functions to filter data.

Example:

- Retrieve orders with specific conditions:

```
SELECT * FROM Orders

WHERE (Amount > 500 OR OrderDate = '2024-01-15') AND CustomerID = 5;
```

Sorting and Filtering Data through Queries

Sorting Data: Use `ORDER BY` to sort results by one or more columns.

Filtering Data: Use `WHERE` to filter rows based on conditions.

Example:

- **Sort Orders by Amount:**

```
SELECT * FROM Orders

ORDER BY Amount DESC;
```

Filter Orders by Date Range:

```
SELECT * FROM Orders

WHERE OrderDate BETWEEN '2023-01-01' AND '2023-12-31';
```

Data Entry Forms

Importance of Data Entry Forms

- **Purpose:** Data entry forms are critical for accurate and efficient data collection.
- **Benefits:**
  - Ensures data consistency and accuracy.
  - Simplifies the data input process for users.
  - Reduces errors by guiding the user through form fields.

Creating Forms for Primary Tables

Primary Tables: Forms should be designed to enter data into key tables such as `Customers`, `Orders`, `Products`.

Form Design: Simple and user-friendly with essential fields (e.g., text fields, dropdowns).

Example: Customer entry form with fields: `CustomerName`, `Email`, `PhoneNumber`.

Implementing Validation Routines

- **Validation Routines:**
  - **Input Masks:** Format input, such as phone numbers or dates.
  - **Completeness Checks:** Ensure that mandatory fields are filled.
- **Examples of Input Masks:**
  - **Phone Number:** `(999) 999-9999`
  - **Date:** `MM/DD/YYYY`
- **Completeness Check Example:**
  - Ensure email field is filled before form submission.

Adding Visual Prompts (Dropdowns, Combo Boxes)

Dropdowns and Combo Boxes:

- Provide users with a predefined list of options, improving data consistency.
- Useful for fields like country selection, product categories, or status types.

Example:

- A dropdown for selecting the country in a customer form:
  - Options: `USA`, `Canada`, `UK`

Ensuring Data Integrity through Well-Designed Forms

- **Ensuring Data Integrity:**
  - Use validation and visual prompts to guide users.
  - Implement constraints at the database level to enforce data consistency.
  - Example: Use primary and foreign keys to enforce relationships and avoid orphan records.

Set Up MSSQL on Docker

1. **Pull the MSSQL Server Docker image:**

docker pull mcr.microsoft.com/mssql/server

Run the MSSQL container:

docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=YourStrong@Passw0rd' -p 1433:1433 -d mcr.microsoft.com/mssql/server

Verify the container is running:

docker ps

Connect Azure Data Studio to MSSQL on Docker

1. **Install Azure Data Studio** if you haven't already:
    - Download from Azure Data Studio.
2. **Open Azure Data Studio** and click on **"New Connection"**.
    - **Server:** `localhost`
    - **Authentication Type:** SQL Login
    - **Username:** `sa`
    - **Password:** `YourStrong@Passw0rd`
    - **Port:** 1433
3. **Connect** to the SQL Server running on Docker.

Creating Tables and Defining Relationships

1. **Create the `Customers` Table:** In Azure Data Studio, open a new query window and run the following SQL script:

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(15)
);
```

Create the `Orders` Table with a Foreign Key Relationship to `Customers`:

```
CREATE TABLE Orders (
        OrderID INT PRIMARY KEY,
        OrderDate DATE NOT NULL,
        Amount DECIMAL(10, 2),
        CustomerID INT,
        FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

Defining Fields and Attributes

- **Primary Key:** `CustomerID` in the `Customers` table.
- **Foreign Key:** `CustomerID` in the `Orders` table links to the `Customers` table.
- **Constraints:**
    - `NOT NULL` ensures fields like `Name` and `OrderDate` cannot be empty.
    - `UNIQUE` ensures there are no duplicate emails.

Importing Data from External Sources

1. **Prepare Data for Import:** Assume we have a CSV file named `customers.csv`

CustomerID,Name,Email,PhoneNumber

1,John Doe,johndoe@example.com,555-1234

2,Jane Smith,janesmith@example.com,555-5678

Import Data Using the SQL Server Import Wizard:

- In **Azure Data Studio**, right-click on your database in the **Connections** pane and select **Import Wizard**.
- Select the **CSV** file as the source and map the columns from the CSV file to the `Customers` table.
- Complete the import process.

Populating Tables with Initial Data

1. **Insert Data Manually:** You can also insert data directly into tables using SQL `INSERT` commands:

```
INSERT INTO Customers (CustomerID, Name, Email, PhoneNumber)
VALUES (1, 'John Doe', 'johndoe@example.com', '555-1234');
```

Bulk Insert: Alternatively, use a bulk insert for larger datasets:

```
BULK INSERT Customers
FROM 'C:/path-to-your-file/customers.csv'
WITH (
        FIRSTROW = 2,
        FIELDTERMINATOR = ',',
        ROWTERMINATOR = '\n'
);
```

Query Data

After populating the tables, you can run queries to verify the data and relationships:

1. **Join `Customers` and `Orders` Tables:**

```
SELECT Customers.Name, Orders.OrderID, Orders.OrderDate, Orders.Amount
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

This query will return customer names and the corresponding orders they have placed.

Manage Your Database with Azure Data Studio

1. **Monitor Performance:**
    - Use Azure Data Studio's built-in **dashboard** to monitor database performance, query execution times, and resource usage.
2. **Run Advanced Queries:**
    - Utilize the query editor to write and execute more complex queries, test data flows, and check for errors in data relationships.