6. Relational Database Benefits

Relational Database Benefits

Relational Database: A type of database that organizes data into related tables using primary and foreign keys.

Key Benefits:

1. **Reduced Data Redundancy**: Data is not duplicated; related data is stored in different tables and linked using foreign keys.
2. **Efficient Data Storage and Retrieval**: Relational databases store data efficiently, reducing wasted space and speeding up retrieval.
3. **Faster Access to Records**: Queries retrieve data faster through indexing and optimized relationships.
4. **Easier Data Manipulation and Reporting**: SQL allows powerful manipulation of data, and relational structures simplify reporting.

Reduced Data Redundancy

- **Redundancy**: Storing duplicate data across multiple locations.
- **How It's Reduced**:
  - Data is divided into smaller, related tables, each storing a specific entity.
  - Relationships are maintained using **primary keys** and **foreign keys**.

Example:

- **Customers Table**: Stores customer information.
- **Orders Table**: Stores order details and links to customers with a **CustomerID** foreign key.

Efficient Data Storage and Retrieval

Efficient Storage:

- Structured data prevents duplication and saves storage space.
- Tables are normalized to eliminate unnecessary repetition.

Efficient Retrieval:

- Relational databases use **SQL** for fast retrieval, allowing the use of conditions, joins, and filters to fetch data from multiple related tables.

Faster Access to Records

Indexing:

- Indexes on primary keys and commonly searched columns speed up data retrieval by avoiding full table scans.

Relationships:

- Data from multiple tables is combined through joins, providing faster results without needing redundant data storage.

Example:

- Fetch all orders for a specific customer by joining the `Customers` and `Orders` tables using the **CustomerID** foreign key.

Easier Data Manipulation and Reporting

Data Manipulation:

- SQL provides robust tools for updating, deleting, or inserting data across multiple related tables.

Reporting:

- Relational databases simplify complex reports by linking tables and allowing data aggregation through SQL queries.

Relational Database Example

- **Entities**: Real-world objects stored in tables.
- **Attributes**: Data stored in the columns of a table.
- **Primary Key**: A unique identifier for each record in a table.
- **Foreign Key**: A reference to a primary key in another table to establish relationships.

Example:

- **Student Table**: `StudentID` (Primary Key).
- **Course Table**: `CourseID` (Primary Key).
- **Enrollment Table**: Links `StudentID` and `CourseID` using foreign keys.

Setting Up Docker MSSQL on Azure

Step 1: Run MSSQL in Docker on Azure.

- Pull and run the MSSQL Docker image:

```
docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=YourPassword!' -p 1433:1433 --name mssql -d mcr.microsoft.com/mssql/server:2022-latest
```

Step 2: Connect to MSSQL from Azure Data Studio or SQL Server Management Studio (SSMS).

- Host: `localhost`
- Port: `1433`
- Username: `sa`
- Password: Use the password set in the Docker command.

Creating Relational Tables with Primary and Foreign Keys

Create the `Students` Table:

```
CREATE TABLE Students (
        StudentID INT PRIMARY KEY IDENTITY(1,1),
        FirstName NVARCHAR(50),
        LastName NVARCHAR(50),
        EnrollmentDate DATE
);
```

Create the `Courses` Table:

```
CREATE TABLE Courses (
        CourseID INT PRIMARY KEY IDENTITY(1,1),
        CourseName NVARCHAR(100)
);
```

Create the `Enrollments` Table with Foreign Keys:

```
CREATE TABLE Enrollments (
        EnrollmentID INT PRIMARY KEY IDENTITY(1,1),
        StudentID INT FOREIGN KEY REFERENCES Students(StudentID),
        CourseID INT FOREIGN KEY REFERENCES Courses(CourseID),
        EnrollmentDate DATE
);
```

Querying the Relational Database

- Use a **JOIN** query to fetch data from related tables.

Example Query:

```
SELECT Students.FirstName, Students.LastName, Courses.CourseName
FROM Enrollments
JOIN Students ON Enrollments.StudentID = Students.StudentID
JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
```

This query retrieves students' names and the courses they are enrolled in by joining the `Students`, `Courses`, and `Enrollments` tables.

Inserting and Testing Data

Insert Data into the `Students` and `Courses` tables:

INSERT INTO Students (FirstName, LastName, EnrollmentDate)

VALUES ('John', 'Doe', '2024-09-01');

INSERT INTO Courses (CourseName)

VALUES ('Database Design');

Insert Data into the `Enrollments` table:

INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate)

VALUES (1, 1, '2024-09-05');

Query the `Enrollments` table:

SELECT * FROM Enrollments;