ON DELETE CASCADE & ON UPDATE CASCADE

In, **ON DELETE CASCADE and ON UPDATE CASCADE** are referential actions you can define for foreign key constraints. These actions determine what happens to the rows in a child table when corresponding rows in the parent table are deleted or updated.

ON DELETE CASCADE

- **Explanation**: When you define a foreign key constraint with ON DELETE CASCADE, it means that if a row in the parent table is deleted, all related rows in the child table will also be deleted automatically.

- **Use Case**: This is useful when the child rows depend entirely on the existence of the parent row, such as appointments related to a patient. If a patient is removed from the database, their appointments should also be removed to maintain data integrity.

ON UPDATE CASCADE

- **Explanation**: When you define a foreign key constraint with ON UPDATE CASCADE, it means that if the primary key in the parent table is updated, the foreign keys in the child table will automatically be updated to match the new primary key value.

- **Use Case**: This is useful when primary keys might change and you want to ensure that related rows in child tables remain consistent. For example, if a doctor's ID is updated, all references to that doctor in other tables should also be updated to maintain referential integrity.

Example Scenario

Consider two tables: Patients and Appointments. The Appointments table has a foreign key that references the Patients table.

Patients Table

```sql
CREATE TABLE Patients (
    PatientId INT PRIMARY KEY,
    PatientName VARCHAR(100) NOT NULL,
    DateOfBirth DATE NOT NULL,
    Gender CHAR(1) DEFAULT 'U',
    Address VARCHAR(255) NOT NULL,
    PhoneNumber VARCHAR(15),
    Email VARCHAR(100) UNIQUE
);

CREATE TABLE Doctors (
    DoctorId INT PRIMARY KEY,
    DoctorName VARCHAR(100) NOT NULL,
    Specialization VARCHAR(100) NOT NULL,
    PhoneNumber VARCHAR(15) NOT NULL,
    Email VARCHAR(100) UNIQUE
);
```

Appointments Table with ON DELETE CASCADE and ON UPDATE CASCADE

```sql
CREATE TABLE Appointments (
    AppointmentId INT PRIMARY KEY,
    AppointmentDate DATE,
    AppointmentTime TIME NOT NULL,
    Status VARCHAR(20),
    PatientId INT,
    DoctorId INT,
    FOREIGN KEY (PatientId) REFERENCES Patients(PatientId) ON DELETE CASCADE,
    FOREIGN KEY (DoctorId) REFERENCES Doctors(DoctorId) ON UPDATE CASCADE
);

-- Inserting records into Patients table
INSERT INTO Patients (PatientId, PatientName, DateOfBirth, Gender, Address, PhoneNumber, Email) VALUES
(1, 'Rahul Kumar', '1992-08-05', 'M', '123 Nehru Street, New Delhi', '9876543210', 'rahul.kumar@example.com'),
(2, 'Anjali Sharma', '1988-03-12', 'F', '456 Mahatma Gandhi Road, Mumbai', '8765432109', 'anjali.sharma@example.com');

INSERT INTO Doctors (DoctorId, DoctorName, Specialization, PhoneNumber, Email) VALUES
(1, 'Dr. Priya Singh', 'Dermatology', '9988776655', 'priya.singh@example.com'),
(2, 'Dr. Arjun Patel', 'Orthopedics', '8877665544', 'arjun.patel@example.com');

-- Inserting records into Appointments table
INSERT INTO Appointments (AppointmentId, AppointmentDate, AppointmentTime, Status, PatientId, DoctorId) VALUES
(1, '2024-08-01', '14:00:00', 'scheduled', 1, 1),
(2, '2024-08-02', '15:30:00', 'scheduled', 2, 2);
```

What Happens?

1. **ON DELETE CASCADE**:

    - If you execute DELETE FROM Patients WHERE PatientId = 1;, all appointments for the patient with PatientId = 1 will also be deleted from the Appointments table.

```
DELETE FROM Patients WHERE PatientId = 2;
```

Verify

```
SELECT * FROM Appointments WHERE PatientId = 2;
```

2. **ON UPDATE CASCADE**:

- If you execute UPDATE Doctors SET DoctorId = 3 WHERE DoctorId = 1;, all references to DoctorId = 1 in the Appointments table will be automatically updated to DoctorId = 3.

  UPDATE Doctors SET DoctorId = 4 WHERE DoctorId = 2;

  Verify
  SELECT * FROM Appointments WHERE DoctorId = 4;

Advantages and Considerations

- **Advantages**:

  - **Automatic Maintenance of Referential Integrity**: These actions automatically maintain the consistency of related data across multiple tables.
  - **Simplification of Statements**: You don't need to write additional statements to delete or update related rows manually.

- **Considerations**:

  - **Cascading Deletes**: Use ON DELETE CASCADE carefully. If not used judiciously, it can lead to accidental deletion of large amounts of related data.
  - **Performance**: Cascading updates or deletes can impact database performance, especially if there are many related rows or complex relationships.

In summary, ON DELETE CASCADE and ON UPDATE CASCADE are powerful tools for maintaining referential integrity in relational databases, ensuring that related data remains consistent automatically.