Implementing Cascading Changes and Automating Updates with Triggers

Triggers can be used to automate updates and enforce cascading changes across multiple tables in a database. This ensures that related data remains consistent and synchronized.

Example: Cascading Updates

Suppose we have two tables: departments and employees, where each employee belongs to a department. If the department name changes, we want to log this change.

Step 1: Create the Departments Table

```
CREATE TABLE departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50)
);
```

Step 2: Create the Employees Table

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    name VARCHAR(50),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

Step 3: Create a Log Table

```
CREATE TABLE department_changes_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    department_id INT,
    old_name VARCHAR(50),
    new_name VARCHAR(50),
    change_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Step 4: Define the Trigger for Cascading Updates

```
CREATE TRIGGER log_department_name_change
AFTER UPDATE ON departments
FOR EACH ROW
BEGIN
```

```
    IF OLD.department_name != NEW.department_name THEN
      INSERT INTO department_changes_log (department_id, old_name, new_name)
      VALUES (OLD.department_id, OLD.department_name, NEW.department_name);
    END IF;
END;
```

Enabling/Disabling Triggers

Sometimes, you may need to enable or disable triggers temporarily, especially during bulk data operations.

Enable a Trigger

```
ALTER TABLE table_name ENABLE TRIGGER trigger_name;
```

Disable a Trigger

```
ALTER TABLE table_name DISABLE TRIGGER trigger_name;
```

Trigger Execution Order

When multiple triggers are defined on the same table for the same event, the order of execution can be important.

MySQL and PostgreSQL

MySQL and PostgreSQL execute triggers in the order they are created. To control the execution order, you can drop and recreate triggers in the desired order.

SQL Server

```
EXEC sp_settriggerorder
    @triggername = 'trigger_name',
    @order = 'FIRST' | 'LAST',
    @stmttype = 'INSERT' | 'UPDATE' | 'DELETE';
```