01.2 CASCADE -KirkYagami

08 Referential Integrity

01.1 Referential Integrity Full Code

Understanding CASCADE in SQL

Introduction to CASCADE

The `CASCADE` keyword in SQL is used with foreign keys to ensure that changes in the parent table are automatically propagated to the child table. It is commonly used in two scenarios:

1. **ON DELETE CASCADE**: When a row in the parent table is deleted, the corresponding rows in the child table are also deleted.
2. **ON UPDATE CASCADE**: When a row in the parent table is updated, the corresponding rows in the child table are also updated.

Benefits of CASCADE

- **Data Integrity**: Ensures referential integrity by automatically handling changes in the parent-child relationship.
- **Ease of Maintenance**: Simplifies the maintenance of related tables by automatically managing related rows.
- **Consistency**: Keeps related data consistent across multiple tables.

Example: Using CASCADE with Foreign Keys

Scenario

We have two tables: `users` and `orders`. The `orders` table has a foreign key referencing the `users` table. We want to ensure that if a user is deleted, all their orders are also deleted.

Table Creation with CASCADE

1. **Table 1:** `users`

```sql
CREATE TABLE users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2. **Table 2:** `orders`

```sql
CREATE TABLE orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    order_date DATE NOT NULL,
    amount DECIMAL(10, 2) NOT NULL CHECK (amount > 0),
    user_id INT,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Explanation of CASCADE Usage

1. **ON DELETE CASCADE**:
   - If a row in the `users` table is deleted, all rows in the `orders` table that reference the deleted `user_id` will also be automatically deleted.
   - This maintains referential integrity by ensuring there are no orphaned records in the `orders` table.
2. **ON UPDATE CASCADE**:
   - If the `user_id` in the `users` table is updated, the corresponding `user_id` in the `orders` table will also be automatically updated.
   - This ensures consistency of the `user_id` value across both tables.

Practical Example

1. **Inserting Data**

```sql
INSERT INTO users (username, email) VALUES ('john_doe', 'john@example.com');
INSERT INTO orders (order_date, amount, user_id) VALUES ('2023-01-01', 100.00, 1);
```

2. **Deleting a User**

```sql
DELETE FROM users WHERE user_id = 1;
```

- Before the delete operation, the `orders` table has a record with `user_id = 1`.
- After the delete operation, the corresponding row in the `orders` table is also deleted because of the `ON DELETE CASCADE` constraint.

3. **Updating a User ID**

```sql
UPDATE users SET user_id = 2 WHERE user_id = 1;
```

- Before the update operation, the `orders` table has a record with `user_id = 1`.

- After the update operation, the `user_id` in the `orders` table is automatically updated to `2` because of the `ON UPDATE CASCADE` constraint.

Summary

- **CASCADE** in SQL ensures automatic propagation of changes from the parent table to the child table.
- **ON DELETE CASCADE**: Automatically deletes rows in the child table when the corresponding row in the parent table is deleted.
- **ON UPDATE CASCADE**: Automatically updates rows in the child table when the corresponding row in the parent table is updated.
- Using `CASCADE` helps maintain data integrity and consistency, making database maintenance easier.

Final Notes

- Use `CASCADE` judiciously as it can lead to unintended deletions or updates if not properly managed.
- Always ensure that the use of `CASCADE` aligns with the business logic and requirements of your application.

By understanding and applying `CASCADE` constraints, you can create more robust and reliable database schemas that handle parent-child relationships effectively.