SQL DELETE-Anweisung

Die `DELETE`-Anweisung in SQL wird verwendet, um Datensätze aus einer Tabelle zu löschen. Sie können einzelne Zeilen, mehrere Zeilen oder alle Zeilen in einer Tabelle löschen. Es ist wichtig, die `DELETE`-Anweisung mit Bedacht zu verwenden, insbesondere wenn keine `WHERE`-Klausel angegeben ist, da dies alle Zeilen in der Tabelle löscht.

Grundsyntax

```
DELETE FROM table_name
WHERE condition;
```

- **`table_name`**: Der Name der Tabelle, aus der die Datensätze gelöscht werden sollen.
- **`condition`**: Eine Bedingung, die bestimmt, welche Zeilen gelöscht werden. Ohne die `WHERE`-Klausel werden alle Zeilen in der Tabelle gelöscht.

Beispiele

Beispiel 1: Löschen eines spezifischen Datensatzes

Angenommen, wir haben eine Tabelle `products`:

id	name	category_id
1	Harley Davidson Sportster	2
2	Boeing 747	3
3	Titanic	4
4	Ferrari	5

Um das Produkt mit der `id` 1 (Harley Davidson Sportster) zu löschen, verwenden wir:

```
DELETE FROM products
WHERE id = 1;
```

Ergebnis:

id	name	category_id
2	Boeing 747	3
3	Titanic	4
4	Ferrari	5

Beispiel 2: Löschen mehrerer Datensätze basierend auf einer Bedingung

Angenommen, wir möchten alle Produkte löschen, die zur Kategorie `4` (Ships) gehören:

```
DELETE FROM products
WHERE category_id = 4;
```

Ergebnis:

id	name	category_id
2	Boeing 747	3
4	Ferrari	5

Beispiel 3: Löschen aller Datensätze in einer Tabelle

Um alle Datensätze in der Tabelle `products` zu löschen (verwenden Sie diese Anweisung mit Vorsicht), verwenden wir:

```
DELETE FROM products;
```

Ergebnis:

id	name	category_id

(keine Zeilen)

Beispiel 4: Löschen eines Datensatzes mit einer komplexen Bedingung

Angenommen, wir möchten alle Produkte löschen, deren Namen das Wort "Boeing" enthalten und die in der Kategorie `3` (Planes) sind:

```
DELETE FROM products
WHERE name LIKE '%Boeing%' AND category_id = 3;
```

Ergebnis:

id	name	category_id
4	Ferrari	5

Wichtige Hinweise

- **Referenzielle Integrität**: Stellen Sie sicher, dass das Löschen von Datensätzen keine referenzielle Integrität verletzt, insbesondere wenn Fremdschlüssel beteiligt sind. Verwenden Sie gegebenenfalls die

`ON DELETE CASCADE`-Option, wenn Sie Fremdschlüssel definieren.

Erklärung von `ON DELETE CASCADE` in SQL

Die `ON DELETE CASCADE`-Option wird in SQL verwendet, um referenzielle Integrität zwischen Tabellen aufrechtzuerhalten. Wenn ein Datensatz in der übergeordneten Tabelle gelöscht wird, bewirkt `ON DELETE CASCADE`, dass automatisch die zugehörigen Datensätze in der untergeordneten Tabelle ebenfalls gelöscht werden.

Grundsyntax

```
CREATE TABLE parent (
    id INT PRIMARY KEY
);

CREATE TABLE child (
    id INT PRIMARY KEY,
    parent_id INT,
    FOREIGN KEY (parent_id) REFERENCES parent(id) ON DELETE CASCADE
);
```

Beispiel mit `ON DELETE CASCADE`

Angenommen, wir haben zwei Tabellen: `categories` und `products`, wobei jedes Produkt einer Kategorie zugeordnet ist. Wenn eine Kategorie gelöscht wird, sollen alle zugehörigen Produkte ebenfalls gelöscht werden.

Erstellung der Tabellen mit `ON DELETE CASCADE`

```
-- Kategorie-Tabelle anlegen
CREATE TABLE categories (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200)
);

-- Produkttabelle anlegen (1-N zu Kategorie)
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE CASCADE
);
```

Verwendung von `ON DELETE CASCADE`

Wenn wir jetzt eine Kategorie löschen, werden alle zugehörigen Produkte ebenfalls gelöscht. Beispielsweise, wenn wir die Kategorie `Classic Cars` löschen, werden die Produkte `Titanic` und `Ferrari` ebenfalls gelöscht.

```
-- Kategorie 'Classic Cars' (id = 1) löschen
DELETE FROM categories
WHERE id = 1;
```

Überprüfung

```
-- Überprüfen der verbleibenden Produkte
SELECT * FROM products;
```

Ergebnis:

id	name	category_id
1	Harley Davidson	2
2	Boeing 747	3

Alternativen zu `ON DELETE CASCADE`

Es gibt verschiedene Alternativen zu `ON DELETE CASCADE`, die verwendet werden können, um die referenzielle Integrität in einer SQL-Datenbank zu gewährleisten. Diese Optionen legen fest, wie sich eine Datenbank verhalten soll, wenn ein Datensatz, auf den von einem Fremdschlüssel verwiesen wird, gelöscht oder aktualisiert wird.

Hier sind die wichtigsten Optionen:

1. **`ON DELETE RESTRICT` / `ON UPDATE RESTRICT`**:
    - Verhindert das Löschen oder Aktualisieren des Datensatzes in der übergeordneten Tabelle, wenn es abhängige Datensätze in der untergeordneten Tabelle gibt. Dies ist die Standardverhaltensweise.
2. **`ON DELETE NO ACTION` / `ON UPDATE NO ACTION`**:
    - Ähnlich wie `RESTRICT`, es verhindert das Löschen oder Aktualisieren des Datensatzes in der übergeordneten Tabelle, wenn es abhängige Datensätze in der untergeordneten Tabelle gibt. Der Unterschied zu `RESTRICT` besteht darin, dass `NO ACTION` von der SQL-Standardisierungsorganisation definiert ist, während `RESTRICT` von einigen Datenbanksystemen wie MySQL bereitgestellt wird.
3. **`ON DELETE SET NULL` / `ON UPDATE SET NULL`**:
    - Setzt den Fremdschlüsselwert in der untergeordneten Tabelle auf `NULL`, wenn der referenzierte Datensatz in der übergeordneten Tabelle gelöscht oder aktualisiert wird. Der Fremdschlüssel

muss nullable sein.

4. `ON DELETE SET DEFAULT` / `ON UPDATE SET DEFAULT`:
    - Setzt den Fremdschlüsselwert in der untergeordneten Tabelle auf einen Standardwert, wenn der referenzierte Datensatz in der übergeordneten Tabelle gelöscht oder aktualisiert wird. Der Standardwert muss vorher definiert sein.

Beispiel: Alternativen zu CASCADE

Verwendung von `RESTRICT`

```
-- Produkttabelle anlegen (1-N zu Kategorie) mit RESTRICT
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE RESTRICT
);
```

Verwendung von `NO ACTION`

```
-- Produkttabelle anlegen (1-N zu Kategorie) mit NO ACTION
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE NO ACTION
);
```

Verwendung von `SET NULL`

```
-- Produkttabelle anlegen (1-N zu Kategorie) mit SET NULL
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE SET NULL
);
```

Verwendung von `SET DEFAULT`

```
-- Produkttabelle anlegen (1-N zu Kategorie) mit SET DEFAULT
CREATE TABLE products (
```

```
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    category_id INT DEFAULT 1, -- Standardwert festlegen
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE SET DEFAULT
);
```

Erklärung der Optionen

- `RESTRICT` und `NO ACTION`: Diese Optionen verhindern das Löschen oder Aktualisieren des übergeordneten Datensatzes, wenn abhängige Datensätze existieren. Der Unterschied liegt hauptsächlich in der Implementierung und Konformität mit verschiedenen SQL-Standards.
- `SET NULL`: Diese Option setzt den Fremdschlüsselwert in der untergeordneten Tabelle auf `NULL`, wenn der übergeordnete Datensatz gelöscht wird. Dies erfordert, dass die Fremdschlüsselspalte NULL-Werte zulässt.
- `SET DEFAULT`: Diese Option setzt den Fremdschlüsselwert in der untergeordneten Tabelle auf einen vordefinierten Standardwert, wenn der übergeordnete Datensatz gelöscht wird. Dies erfordert, dass ein Standardwert für die Fremdschlüsselspalte definiert ist.

Alternativen zu `ON DELETE CASCADE` in SQL

Verwendung von `RESTRICT`

```
-- Produkttabelle anlegen (1-N zu Kategorie) mit RESTRICT
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE RESTRICT
);
```

Verwendung von `NO ACTION`

```
-- Produkttabelle anlegen (1-N zu Kategorie) mit NO ACTION
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE NO ACTION
);
```

Verwendung von `SET NULL`

```sql
-- Produkttabelle anlegen (1-N zu Kategorie) mit SET NULL
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE SET NULL
);
```

Verwendung von `SET DEFAULT`

```sql
-- Produkttabelle anlegen (1-N zu Kategorie) mit SET DEFAULT
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    category_id INT DEFAULT 1, -- Standardwert festlegen
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE SET DEFAULT
);
```

Fazit

- `ON DELETE RESTRICT` und `ON DELETE NO ACTION`: Verhindern das Löschen von übergeordneten Datensätzen, wenn abhängige Datensätze existieren.
- `ON DELETE SET NULL`: Setzt den Fremdschlüsselwert in der untergeordneten Tabelle auf `NULL`, wenn der übergeordnete Datensatz gelöscht wird.
- `ON DELETE SET DEFAULT`: Setzt den Fremdschlüsselwert in der untergeordneten Tabelle auf einen vordefinierten Standardwert, wenn der übergeordnete Datensatz gelöscht wird.
- `ON DELETE CASCADE`: Löscht automatisch die abhängigen Datensätze, wenn der übergeordnete Datensatz gelöscht wird.