Lernskript: SQL-Grundlagen & JDBC

Relationale Datenbanken sind das Rückgrat der meisten Geschäftsanwendungen. SQL ist die Sprache, um mit ihnen zu sprechen, und JDBC ist die Brücke von Java zur Datenbank.

Teil 1: Die grundlegende Analogie - Das "Warum"

- **Relationale Datenbank (z.B. MySQL):** Stell sie dir wie einen Aktenschrank voller perfekt organisierter Tabellen vor.
    - **Tabelle:** Ein Ordner (z.B. `BENUTZER`).
    - **Zeile:** Ein Datensatz (z.B. ein spezifischer Benutzer).
    - **Spalte:** Ein Attribut (z.B. `ID`, `NAME`).
- **SQL (Structured Query Language):** Die universelle Sprache, um Anfragen an diesen Aktenschrank zu stellen ("Gib mir...", "Speichere...", "Ändere...").
- **JDBC (Java Database Connectivity):** Die **Telefonleitung und der Übersetzer** zwischen deiner Java-Anwendung und der Datenbank. Es ist eine Java-Standard-API, um SQL-Befehle zu senden und Antworten zu empfangen.

Teil 2: Die 4 CRUD-Operationen mit SQL

CRUD steht für die vier fundamentalen Operationen mit Daten: **C**reate, **R**ead, **U**pdate, **D**elete.

- **1. `CREATE` -> `INSERT`**

  Fügt einen neuen Datensatz (eine neue Zeile) in eine Tabelle ein.

```
INSERT INTO BENUTZER (ID, NAME, EMAIL) VALUES (1, 'Anna',
'anna@example.com');


  2. READ -> SELECT

  Liest Daten aus einer oder mehreren Tabellen. Das ist der häufigste
Befehl.
  Generated sql
```

-- Lese alle Spalten (*) aller Benutzer SELECT * FROM BENUTZER;

-- Lese nur den Benutzer mit einer bestimmten ID SELECT * FROM BENUTZER WHERE ID = 1;

-- Lese verknüpfte Daten aus zwei Tabellen SELECT * FROM BESTELLUNGEN B JOIN BENUTZER U ON B.BENUTZER_ID = U.ID;

3. UPDATE

Ändert bestehende Datensätze in einer Tabelle. Generated sql

UPDATE BENUTZER SET EMAIL = 'neue.email@example.com' WHERE ID = 1;

4. DELETE

Löscht Datensätze aus einer Tabelle. Generated sql

DELETE FROM BENUTZER WHERE ID = 1;

Teil 3: Vertiefung (JavaMasta's Profi-Tipps)

```
JDBC ist "Low-Level": Die direkte Nutzung von JDBC erfordert viel Code
(Verbindungen öffnen/schließen, manuelle Verarbeitung von Ergebnissen). In der
modernen Entwicklung wird es selten direkt verwendet.

Abstraktionen sind der Schlüssel: Moderne Frameworks wie Spring Data JPA (Thema in
Phase 3) bauen auf JDBC auf und nehmen dem Entwickler fast die gesamte Arbeit ab.
Statt SQL zu schreiben, ruft man Java-Methoden wie repository.save(benutzer) auf.

SQL-Injection (Die größte Gefahr): Niemals, unter gar keinen Umständen, SQL-
Befehle durch das Zusammenfügen von Benutzereingaben und Strings bauen! Dies
öffnet ein massives Sicherheitsloch.

PreparedStatement (Die Lösung): Der sichere Weg in JDBC, um SQL-Injection zu
verhindern. Der Befehl wird mit Platzhaltern (?) an die Datenbank gesendet, und
die Daten werden separat und sicher eingefügt.
Generated java
```

// Sicherer Weg mit JDBC String sql = "SELECT * FROM BENUTZER WHERE ID = ?"; PreparedStatement stmt = connection.prepareStatement(sql); stmt.setInt(1, benutzerId); // Setzt den Wert sicher in den ersten Platzhalter