6.4 Daten bearbeiten mit SQL

SQL bietet als Data Manipulation Language (DML) die Möglichkeit, Daten zu bearbeiten, indem neue Datensätze eingefügt, bestehende gelöscht oder geändert werden können.

6.4.1 Einfügen von Datensätzen

Um die Datenbestände entsprechend den laufenden Geschäftsprozessen zu aktualisieren, werden neue Daten hinzugefügt und überflüssige Daten gelöscht. Einfügen von Datensätzen geschieht in SQL mit der Schlüsselanweisung `INSERT`.

Die allgemeine Syntax der `INSERT`-Anweisung lautet:

```
INSERT INTO Tabelle (Feld1, Feld2, ...)
VALUES (Inhalt1, Inhalt2, ... );
```

```
INSERT INTO Tabelle (Feld1, Feld2, ...)
SELECT (Feld1, Feld2, ...) ...;
```

Beispiel:

Wir fügen mit einer SQL-Anweisung einen neuen Datensatz für das Tourenrad „Easygo" des Herstellers Stiegl und der Fahrradnummer 12 in die Fahrradtabelle ein.

Lösung:

```
INSERT INTO Fahrraeder (Fahrradnummer, Hersteller, Bezeichnung,
Art)
VALUES (12, ‚Stiegl', ‚Easygo', ‚Tourenrad');
```

Hinweis:

Die Reihenfolge der nach `INSERT` aufgelisteten Felder entspricht den nach `VALUES` aufgelisteten Werten. Der Datentyp der Felder muss gleich dem Datentyp des entsprechenden Wertes sein.

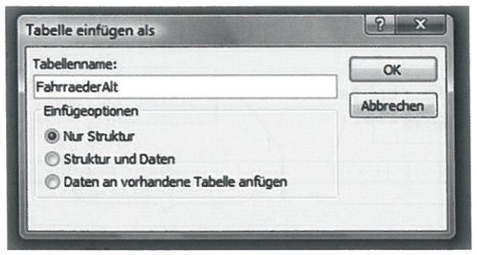
Werden die einzutragenden Daten aus einer bestehenden Tabelle ausgewählt, so wird eine `SELECT`-Anweisung eingebunden. Die Struktur der `INSERT`-Anweisung lautet dann:

Beispiel:

Fahrräder, die vor 2014 angeschafft wurden, sollen ausgemustert werden. Um die Daten nicht endgültig zu verlieren, benötigt man eine Anweisung, die zur Sicherung die Fahrraddaten in eine Tabelle `FahrraederAlt` einfügt.

```
INSERT INTO FahrraederAlt
SELECT *
FROM Fahrraeder
WHERE Anschaffungsdatum<#1/1/2014#;
```

Es wird eine Tabelle `FahrraederAlt` benötigt mit der gleichen Tabellenstruktur. Diese kann man mit einer CREATE TABLE-Anweisung erstellen oder die bestehende Tabelle `Fahrraeder` z. B. in Access kopieren, indem man im Kontextmenü erst `Kopieren` und dann `Einfügen` wählt. Dort wird die Option `Nur Struktur` markiert.



Im Beispiel werden über die SELECT-Anweisung alle Datensätze der Fahrräder gesucht, die vor 2014 angeschafft wurden, und anschließend in die neue Tabelle gespeichert.

Hinweis:

Datentypen und Namen der korrespondierenden Felder der beiden Tabellen müssen bei dieser Methode übereinstimmen.

Beispiel:

Mit einer INSERT-Anweisung sollen die Daten des neuen Kunden Fritz Kleidermann aus 70173 Stuttgart, Mantelstr. 128 und der Kundennummer Kd_Nr = 2658 eingefügt werden.

Lösung:

```
INSERT INTO Kunden (Kd_Nr, Nachname, Vorname, PLZ, Ort, Strasse)
VALUES (2658, 'Kleidermann', 'Fritz', 70173, 'Stuttgart', 'Mantelstr. 128')
```

Hinweis:

Bei dieser Methode des Einfügens definierter Werte mit Hilfe des Schlüsselworts VALUES muss in jedes Feld, das wegen des Attributs NOT NULL nicht leer sein darf, ein Wert eingefügt werden.

6.4.2 Löschen von Datensätzen

Das Löschen der ausgelagerten Datensätze geschieht mit dem SQL-Befehl `DELETE`. Die allgemeine Syntax lautet:

```
DELETE Attributname/* FROM Tabelle
WHERE Bedingung;
```

Beispiel:

Eine SQL-Anweisung soll die Fahrräder, die vor 2014 angeschafft wurden, aus der Tabelle `Fahrraeder` löschen.

Lösung:

```
DELETE *
FROM Fahrraeder
WHERE Anschaffungsdatum < #1/1/2014#;
```

Hinweis:

Der Löschbefehl DELETE kann nicht mehr rückgängig gemacht werden. Deshalb ist es empfehlenswert, zunächst mit dem Befehl SELECT nach den zu löschenden Daten zu suchen. Nach Überprüfung werden die Daten gelöscht.

Empfehlenswert ist stets, eine Sicherungskopie der Daten aufzubewahren.

6.4.3 Aktualisieren von Daten

Zum Verändern von Feldinhalten dient der Aktualisierungsbefehl `UPDATE`, z. B. wenn viele Datensätze gleichsinnig geändert werden sollen.

Die allgemeine Syntax lautet:

```
UPDATE Tabelle
SET Attributname = Wert
WHERE Bedingung;
```