**Datenbankabfragen**

Um Daten aus einer Datenbank abzufragen oder Daten zu manipulieren (hinzufügen, ändern, löschen) benötigt man Anweisungsobjekte, die von der Verbindung (*Connection*) zur Verfügung gestellt werden. Folgende Anweisungsobjekte gibt es:

* *Statement* (normale Anweisung)
* *PreparedStatement* (vorbereitete bzw. vorübersetzte Anweisung)
* *CallableStatement* (Aufruf gespeicherter Datenbank-Prozeduren)

Für SQL-Befehle verwendet man in der Regel das *PreparedStatement*. Es hat folgende entscheidende Vorteile gegenüber dem normalen *Statement*:

* Sicherer, da es eine SQL-Injection verhindert.
* Schneller bei mehrmaligem Aufruf, da der grundlegende SQL-Befehl der Datenbank lediglich einmal bekannt gemacht wird und dann nur noch die ver­än­derlichen Parameter übertragen werden müssen.

Das *PreparedStatement* ermöglicht es also, Platzhalter in einen SQL-Befehl einzubauen, die erst später mit Werten gefüllt werden.

**Beispiele**

Datensätze hinzufügen:

|  |
| --- |
| Connection conn = …;  String sql = "INSERT INTO kunde (vorname, nachname) VALUES (?, ?)";  PreparedStatement statement = conn.prepareStatement(sql);  statement.setString(1, "Hans");  statement.setString(2, "Mustermann");  statement.executeUpdate();  statement.setString(1, "Erika");  statement.setString(2, "Musterfrau");  statement.executeUpdate(); |

Datensatz ändern:

|  |
| --- |
| Connection conn = …;  String sql = "UPDATE kunde SET vorname = ?, nachname = ? WHERE id = ?";  PreparedStatement statement = conn.prepareStatement(sql);  statement.setString(1, "Alfons");  statement.setString(2, "Musterherr");  statement.setInt(3, id); // Die Variable id enthält die ID des Datensatzes.  statement.executeUpdate(); |

Datensatz löschen:

|  |
| --- |
| Connection conn = …;  String sql = "DELETE FROM kunde WHERE id = ?";  PreparedStatement statement = conn.prepareStatement(sql);  statement.setInt(1, id); // Die Variable id enthält die ID des Datensatzes.  statement.executeUpdate(); |

Datensätze abfragen:

|  |
| --- |
| Connection conn = …;  String sql = "SELECT \* FROM kunde WHERE nachname = ?";  PreparedStatement statement = conn.prepareStatement(sql);  statement.setString(1, "Mustermann");  ResultSet resultSet = statement.executeQuery(); |

**ID's ermitteln bei Autowerten**

Wird ein neuer Datensatz in eine Tabelle eingefügt, die als Primärschlüssel ein Attribut besitzt, welches datenbankseitig per Autowert bestimmt wird, ist unter Umständen wichtig, die ID eines gerade hinzugefügten Daten­satzes zu kennen.

Problem:

Im Prinzip könnte man direkt nach dem Hinzufügen eines Datensatzes per Select-Befehl den Datensatz mit der höchsten ID heraussuchen.

Das klappt auch ganz gut, wenn man eine Datenbank voraussetzt, die Anfragen nur von einer einzigen Anwendung aus zulässt. Denn in einer mehrbenutzerfähigen Datenbank lässt sich nicht aus­schließen, dass ein anderer Benutzer gerade ebenfalls einen Datensatz hinzugefügt hat. Eventuell bekäme man also nicht die ID des eigenen Datensatzes, sondern die des Datensatzes des anderen Benutzers.

Eine Lösung dieses Problems per SQL ist also bei mehrbenutzerfähigen Datenbanken so einfach nicht möglich.

Lösung:

JDBC bzw. das *PreparedStatement* bietet dafür eine Lösung - nämlich die Methode *getGeneratedKeys()*, die allerdings nicht in allen JDBC-Treibern implementiert ist.

Beim Treiber für MS Access z.B. ist sie es nicht. Access ist aber auch keine mehrbenutzerfähige Datenbank, also benötigt man diese Variante auch nicht.

Wer allerdings mit einer DB2, MySQL oder SQLite Datenbank arbeitet, kann diese Funktionalität von JDBC nutzen.

Anwendung von *getGeneratedKeys()* bei der DB2:

|  |
| --- |
| Connection conn = …;  String sql = "INSERT INTO kunde (vorname, nachname) VALUES (?, ?)";  PreparedStatement statement = conn.prepareStatement(sql,  Statement.RETURN\_GENERATED\_KEYS);  statement.setString(1, "Hans");  statement.setString(2, "Mustermann");  statement.executeUpdate();  ResultSet resultSet = statement.getGeneratedKeys();  resultSet.next();  int kundenNr = resultSet.getInt(1); |

Anwendung von *getGeneratedKeys()* bei SQLite:

|  |
| --- |
| Connection conn = …;  String sql = "INSERT INTO kunde (vorname, nachname) VALUES (?, ?)";  PreparedStatement statement = conn.prepareStatement(sql);  statement.setString(1, "Hans");  statement.setString(2, "Mustermann");  statement.executeUpdate();  ResultSet resultSet = statement.getGeneratedKeys();  resultSet.next();  int kundenNr = resultSet.getInt("last\_insert\_rowid()"); |