Leitprogramm

Datenbanken

Stand 24. September 2016

Inhaltsverzeichnis

Kapitel 1

Vorwort

Im deutschen Sprachgebrauch wird der Computer oft als Rechner bezeichnet. Doch anstatt mathematische Probleme zu lösen, dient der heimische Computer vielen Menschen als bessere Schreibmaschine und Abspielgerät von digitalen Werken. Die weitere Aufgabe, die Computern zukommt, nehmen viele Menschen nur indirekt wahr: Das Verwalten von Daten. Diese Aufgabe beherrschen Computer so gut, dass sie an immer mehr Stellen zum Einsatz kommen und schon allgemein von einer »Datensammelwut« gesprochen wird.

Daher ist es angebracht, einen Blick hinter die Kulissen dieser Sammler und Verwalter von Daten zu werfen. Es gilt, die Methoden und Werkzeuge kennen zu lernen, sie selber in eigene Programme einzubauen und die Risiken und Auswirkungen auf uns als Person und unsere Gesellschaft abzuschätzen.

Begonnen wird mit einer Problemstellung, die mit bekannten Methoden aus der Objektorientierung analysiert wird. Dafür wird eine eigene Datenbank aufgebaut, mit der gearbeitet werden soll, so dass sich die Problemstellung als roter Faden durch das Thema zieht. Doch zu Beginn gibt es erst ein paar Fragen zur eigenen Einschätzung und zum eigenen Vorwissen:

Aufgaben 1.1

a) Da Datenbanken anscheinend so gegenwärtig sein sollen, sind Sie hier dazu aufgefordert, anzugeben, wo Sie direkt oder indirekt mit Datenbanken in Berührung kommen oder gekommen sind.

b) Erläutern Sie, welche Aufgaben und welchen Zweck die verwendeten Datenbanken

aus den oben aufgezählten Punkten haben.

c) Geben Sie an, wie Sie die Nützlichkeit der Datenbanken einschätzen und für wen dieser Nutzen besteht.

d) Jede Technologie hat auch ihre Schattenseiten. Führen Sie die Gefahren von Datenbanken an, die Sie bereits kennen.

Datenstrukturen und Datenbanken

Im Unterricht wurden verschiedene Datenstrukturen aufgezeigt, mit denen sich Daten verwalten lassen: Lineare Strukturen, wie die Liste oder der Stapel, und komplexere wie Bäume und Graphen. Ähnlich verhält es sich mit Datenbanken. Sie strukturieren Daten und erlauben es, in ihnen die Daten zu suchen, einzufügen oder sie zu löschen.

Aufgaben 1.2

a) Zählen Sie die Punkte auf, in denen sich ihrer Meinung nach die bisherigen Datenstrukturen von Datenbanken unterscheiden.

Kapitel 2

Verwaltung von Schülern

Das König-Joseph-Gymnasium möchte seine Schülerinnen und Schüler nicht mehr nur auf Papier erfassen, sondern endlich dem Fortschritt folgen und dieses elektronisch machen. Bei diesem Schritt sollen auch gleich ein paar andere Sachen mit erfasst werden. Um die Anforderungen besser fassen zu können, haben Sie eine Situation schriftlich festgehalten, die mit der Verwaltung umsetzbar sein muss:

> Die beiden Schülerinnen Eva Meier und Claudia Görtz kommen nach einem Umzug zum 2. Halbjahr an das König-Joseph-Gymnasium. Bei der Erfassung ihrer Daten gibt Eva an, dass sie am 24. Januar 1998 geboren wurde und in der Holzgasse 42 in 98 765 Posbed wohnt. Da sie in die Stufe Q1 geht, wird sie von der Jahrgangsstufenbetreuerin Frau Umardo in Empfang genommen. Diese teilt Eva unter anderem dem Informatikkurs Q1-IF2 bei Frau Petermann, dem Pädagogikkurs Q1-PA1 bei Herrn Küsters und dem Mathematikkurs Q1-M4 bei ihr selbst zu. Außerdem erhält Eva einen Stundenplan. In diesem erkennt sie die Kürzel ihrer Lehrer: Pet, Kue und Uma.
> Auch Claudia gibt ihre Daten an. Sie ist am 17. April 1999 geboren und wohnt nun auf dem Holzpfad 23 in 98 789 Umsstadt. Sie wird vom Stufenbegleiter der Einführungsphase, Herrn Hospital (Hos), begrüßt. Auf ihrem Stundenplan sieht sie, dass sie im Mathematikkurs E-M2 bei Frau Umardo und im Physikkurs E-PH1 von Frau Hellfeuer (Hel) ist.

Um von dieser Situationsbeschreibung über eine Modellierung zu einer konkreten Datenbankstruktur zu gelangen, bedient man sich z. B. des Verfahrens von Abbott, das aus dem Unterricht bekannt ist. Dabei werden aus dem Text Kandidaten für Objekte, Methoden und Attributwerte herausgesucht, um diese anschließend verschiedenen Objekten zuzuordnen.

Hinweis

Neben der Methode von Abbott gibt es auch andere Möglichkeiten, eine Modellierung anzugehen. Sollte Ihnen diese Methode nicht bekannt sein, so wenden Sie ihre Methode an, um an die nötigen Objekte und Klassen zu gelangen, die für eine Realisierung der oberen Situationsbeschreibung nötig sind.

Aufgaben 2.1

a) Beschreiben Sie den genauen Ablauf des Verfahrens von Abbott inklusive der Erstellung der Objektkarten.

b) Füllen Sie die folgende Tabelle mit Kandidaten.

Objekte	Methoden	Attributwerte

c) Erstellen Sie aus den Kandidaten Objektkarten und fügen Sie diese zu einem Objektdiagramm zusammen.

2.1 Sequenzdiagramm

Im nächsten Schritt wird zum Objektdiagramm ein Sequenzdiagramm erstellt, das alle Interaktionen zwischen den Objekten wiedergibt. Damit lässt sich genauer analysieren, ob alle benötigten Methoden und Attribute vorhanden sind und welche Beziehungen die Objekte untereinander eingehen müssen.

Aufgaben 2.2

a) Erstellen Sie für die Situationsbeschreibung ein Sequenzdiagramm. Nutzen Sie dabei das Objektdiagramm als Grundlage, die sie an den notwendigen Stellen ergänzen.

2.2 Klassendiagramm

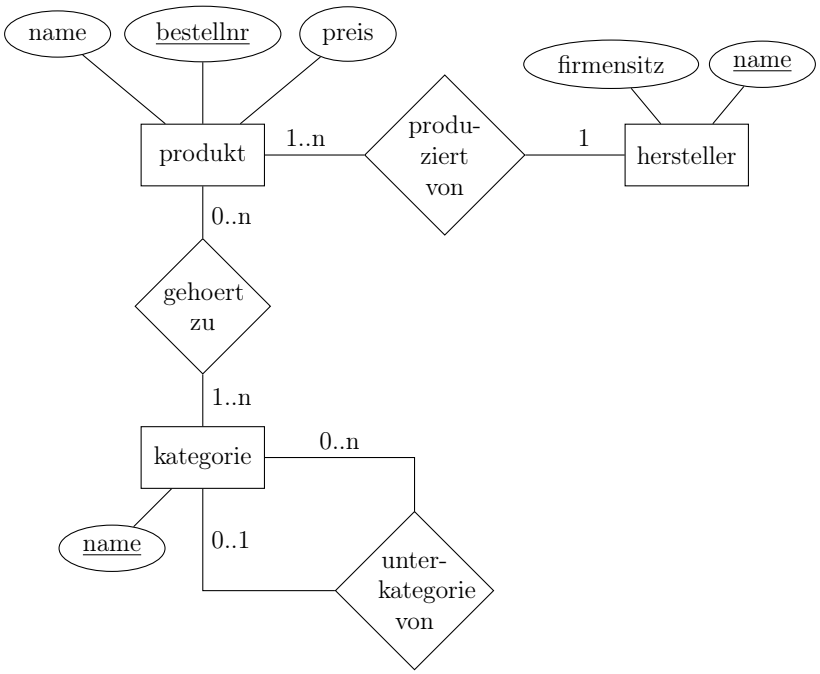
Objekte, die sich in den Methoden und Attributen überschneiden, lassen sich zu Klassen zusammenfassen, die einen gemeinsamen Bauplan darstellen. Zu beachten dabei ist, dass die im Objektdiagramm vorhanden Beziehungen als Beziehungsattribute zwischen den Klassen erhalten bleiben. Dabei gilt es auch, die Anzahl der beteiligten Objekte auf den beiden Seiten einer Beziehung zwischen zwei Klassen zu ermitteln. Diese sind als Kardinalzahlen mit im Diagramm anzugeben.

Aufgaben 2.3

a) Fertigen Sie ein Klassendiagramm auf der Grundlage Ihres Objektdiagramms an, mit der sich die Situationsbeschreibung umsetzen lässt.

2.3 Entity-Relationship-Modell

Das Entity-Relationship-Modell (kurz ER-Modell) ist eine Möglichkeit zur Darstellung der Struktur von Daten. Ein solches Modell wird für Projekte erstellt, damit sie als Grundlage für das Design einer relationalen Datenbank dienen können. Es kann als Vorläufer des Klassenmodells bezeichnet werden, das erst wesentlich später entstanden ist.



Hier dargestellt ist das Beispiel eines ER-Modells für ein Geschäft. Es besteht aus Attributen, Entitätstypen und Beziehungen (Relationships).

2.4 Entitäten

Die Entitäten aus dem ER-Modell sind am besten mit Objekten zu vergleichen. Innerhalb der Zeichnung werden die Entitätstypen in Rechtecken gezeichnet. Die zu einem Entitätstyp gehörenden Attribute (Eigenschaften) stehen in Ellipsen, die mit dem Entitätstyp verbunden sind. Im oberen Beispiel ist also »produkt« ein Entitätstyp mit den Attributen »name«, »bestellnr« und »preis«.

2.5 Beziehungen

Zwischen diesen Entitätstypen sind mit Rauten die Beziehungen eingezeichnet. Wie auch in Klassendiagrammen lassen sich hier Kardinalzahlen angeben. Im oberen Beispiel gibt es die »gehört zu«-Beziehung zwischen Produkt und Kategorie. Dabei ist jedes Produkt mindestens einer Kategorie zugeordnet. Es kann aber auch Kategorien geben, zu denen es kein Produkt gibt. Natürlich sind auch Beziehungen zum gleichen Entitätstyp möglich.

2.6 Schlüssel im ER-Modell

Im Beispiel ist zu erkennen, dass einige Attribute unterstrichen sind. Dieses bedeutet, dass es sich hierbei um Schlüsselattribute handelt. Ein Schlüsselattribut ist dadurch gekennzeichnet, dass sich dadurch eine einzelne Entität von allen anderen Entitäten des gleichen Typs unterscheiden lässt. Die Werte müssen also eindeutig sein. Dieses könnte z. B. bei einem Produkt die Bestellnummer sein, unter der es in einem Laden zu bekommen ist. Außerdem soll für ein solches Schlüsselattribut sichergestellt sein, dass sich dieses Attribut nicht ändert.

Es ist aber auch möglich, dass ein solcher eindeutiger Schlüssel erst durch die Kombination von mehreren Attributen entsteht. So geht man z. B. davon aus, dass die Kombination aus Geburtsname, Vorname, Geburtstag und Geburtsort eindeutig für eine einzelne Person ist, während jedes Attribut für sich alleine auf viele Personen zutreffen würde. Da ein solcher Schlüssel sich nicht in jedem Fall finden lässt, wird häufig ein weiteres Attribut als künstlicher Schlüssel hinzugenommen. Eine Möglichkeit dafür wäre eine fortlaufende Nummer, die dann als Identifikationsnummer oder kurz ID bezeichnet wird.

Aufgaben 2.4

a) Zählen Sie fünf frei wählbare Beispiele für Entitätstypen auf, bei denen Sie die Schlüsselattribute angeben. Unter den Beispielen sollen alle drei Formen (einfacher, zusammengesetzter und künstlicher) Schlüssels vorkommen. Begründen Sie die Wahl des Schlüsselattributs und geben Sie die Schlüsselform mit an.

b) Überführen Sie das im Abschnitt 2.2 erstellt Klassendiagramm für die Verwaltung von Schülerinnen und Schülern in ein ER-Modell.

Kapitel 3

Relationale Datenbanken

Aufgrund des Schemas, wie Daten zusammengefasst und geordnet sind, unterscheidet man die Arten von Datenbanksystemen. Neben relationalen Datenbanksystemen gibt es auch noch objektorientierte und hierarchische Datenbanksysteme. Im Unterricht werden aber nur relationale Datenbanken behandelt, da die Systeme dafür sehr ausgereift und schnell sind. Zusätzlich gibt es die grundlegende Sprache »Structured Query Language« oder kurz SQL, die bei fast allen Datenbankanwendungen Verwendung findet.

Ein Relationenschema setzt sich aus mehreren Attributen zusammen. Eine Relation zu einem Relationenschema ist eine Menge von Tupeln, die zu dem Relationenschema passen. Hier ein Beispiel: Ein Relationenschema zur Speicherung von Kundendaten bekommt den Namen `kunde` und hat die Attribute kundennr, passwort, name, adresse, telefon, geburtsdatum und eMail, wobei kundennr als Schlüssel dient. Das Relationenschema wird wie folgt aufgeschrieben:

```
kunde(kundennr, passwort, name, adresse, telefon, geburtsdatum, eMail)
```

Relationen können auch als Tabellen dargestellt werden. Die Attribute des Relationenschemas bilden bei der Relation die Spaltenüberschriften und die Zeilen sind die Tupel.

Relationenname	**Schlüssel**		**Attribut**		
kunde	kundennr	passwort	name	...	**Relationenschema**
	BT02	saibot	Tobias Bayer	...	
	MK07	nairolf	Knut Maier	...	**Tupel**
	...	...	...	...	

Eine komplette Datenbank setzt sich aus mehreren Relationen bzw. Tabellen zusammen, die jeweils einem Relationenschema entsprechen. Die Relationenschemata bilden dabei das Datenbankschema für die gesamte Datenbank.

Hinter diesem Aufbau durch Relationen gibt es auch eine theoretische Grundlage. Diese wird durch die Relationale Algebra bzw. das Relationenkalkül beschrieben.

3.1 Die Sprache SQL und MySQL

Die Datenbanksprache SQL unterscheidet sich in ihrer Struktur vollkommen von den bisher kennengelernten Programmiersprachen. Daher ist es nötig, eine komplett neue Sprache zu lernen. SQL ist aber im Bereich der relationalen Datenbanken die Standard-Sprache, so dass sie sowohl bei dem hier eingesetzten System MySQL als auch bei anderen relationalen Datanbanksystemen genutzt werden kann. Die möglichen geringfügigen Unterschiede sind in den hier thematisierten Bereichen nicht relevant.

Die Grundbefehle von SQL lassen sich nach drei verschiedenen Aufgabenbereichen sortieren: Die Struktur der Tabellen festlegen, Daten in den Tabellen beeinflussen und Daten aus den Tabellen abfragen. Zur Struktur der Tabellen gehören die Befehle **`CREATE`** (erzeugen), **`ALTER`** (verändern) und **`DROP`** (entfernen). Das Verändern der Daten funktioniert mit **`INSERT INTO`** (einfügen), **`UPDATE`** (aktualisieren) und **`DELETE`** (löschen). Für das Abfragen der Daten gibt es nur den Befehl **`SELECT`** (auswählen), dessen Aufbau aber sehr komplex gestaltet werden kann.

Das Prinzip des Aufbaus von SQL-Befehlen lässt sich an einer einfachen **`SELECT`**-Abfrage verdeutlichen: Es werden die Namen und Preise aller Produkte vom Hersteller Augusto gesucht. Dazu nimmt man die einfachste Form der Abfrage: **`SELECT`** `<spalten>` **`FROM`** `<tabelle>` **`WHERE`** `<bedingung>;`. Für eine Datenbank, die nach dem ER-Modell aus dem Beispiel aufgebaut ist, müsste man nun also die Spalten »name« und »preis« aus der Tabelle »Produkt« wählen. Mit der entsprechenden Bedingung sieht dann die Anfrage so aus: **`SELECT`** `name, preis` **`FROM`** `produkt` **`WHERE`** `hersteller = 'Augusto';`.

Genauso wie in der Sprache Java wird auch in SQL jeder Befehl mit einem Semikolon beendet. Die schrägen Hochkommata sind bei den Spaltenbezeichnern und Tabellen nur dann nötig, wenn diese Sonderzeichen oder Leerzeichen enthalten. Die Zeichenkette in der Bedingung muss aber in jedem Fall mit Hochkommata abgegrenzt werden. Anstatt bei einer größeren Tabelle alle Spalten einzeln aufzuzählen, kann man auch ein Asterix ($*$) verwenden, wie in der folgenden Abfrage: **`SELECT`** `*` **`FROM`** `produkt` **`WHERE`** `hersteller = 'Augusto';`. Der genauere Aufbau, auch der anderen Befehle, wird nun schrittweise erarbeitet.

3.2 Das Hilfsmittel phpMyAdmin

Um mit der Datenbank arbeiten zu können, benötigt man Hilfsmittel, mit denen man auf sie zugreift. Ein übersichtliches aber auch mächtiges Hilfsmittel ist phpMyAdmin, das auf einem Server installiert sein muss und über einen Webbrowser bedient wird. Hiermit kann man sich leicht den Inhalt der Tabellen anzeigen lassen, sowie Tabellen und Daten erstellen, ändern und löschen. Dazu sind keine Kenntnisse von SQL nötig, da die meisten Funktionen sich über einfache Formulare erledigen lassen. Dabei wird nach dem Absenden des Formulars der zur Anweisung gehörende SQL-Befehl angezeigt.

Gleichzeitig bietet phpMyAdmin die Möglichkeit SQL-Befehle selbst anzugeben und auszuführen.

Nach dem Einloggen in phpMyAdmin hat man auf der linken Seite ein Menü, mit dem man die Datenbank und ggf. die Tabelle auswählen kann, mit der man arbeiten möchte. Je nach Auswahl ändert sich der rechte Inhaltsbereich und bietet einem über die oberen Reiter verschiedene fast selbsterklärende Optionen. Im Schulnetzwerk ist phpMyAdmin unter `http://db-server/phpmyadmin` zu erreichen.

Aufgaben 3.1

a) Lassen Sie sich in der Datenbank »sportverein« die Tabellen »mitglieder« und »sportart« anzeigen und schreiben Sie für beide jeweils das Relationenschema auf. L

b) Geben Sie den SQL-Befehl an, mit dem phpMyAdmin die Inhalte von Tabellen anzeigt und erläutern Sie seinen Aufbau. L

3.3 Die Bedienung in der MySQL-Konsole

Neben aufwendigen Oberflächen ist es auch möglich, direkt über die Konsole mit dem Datenbanksystem zu kommunizieren. Dazu dient das Programm mysql, dass auf dem entsprechenden Rechner installiert sein muss. Eine Installationsanleitung dazu können Sie auch im Anhang A finden. Über die Programmparameter lässt sich angeben, mit welchem Datenbanksystem gearbeitet werden soll, welcher Benutzer sich einloggt und auch, welche Datenbank genutzt wird. Der Aufruf mit allen aufgezählten Parametern ist so aufgebaut: `mysql -u <benutzer> -p -D <datenbankname> -h <MYSQL-Server>`. Dieses sieht dann so aus: `mysql -u q2mustermann -p -D q2 -h db-server`. Das `-p` ohne weitere Angabe des Passworts sorgt dafür, dass man dieses geschützt eingeben kann. Wenn man mit der Konsole arbeitet sind folgende Befehle nützlich:

- **`USE`** `<datenbank>;` Wechselt die Datenbank, in der gearbeitet wird.
- **`SHOW TABLES;`** Zeigt an, welche Tabellen in der aktuellen Datenbank vorhanden sind.
- **`DESCRIBE`** `<tabelle>;` Zeigt das Schema der Tabelle an.

Die Konsole wird auch genutzt um viele Befehle auf einmal, wie z. B. bei einem Import durchzuführen. Alle Anweisungen stehen dann in einer Datei, deren Inhalt wird, wie hier im Beispiel angegeben, dem Aufruf mit übergeben: `mysql -u meier -p -D mitglieder -h db-server < neu_mitglieder.sql`.

Aufgaben 3.2

a) Lassen Sie sich das Schema der Tabelle »mitglieder« in der Datenbank »sportverein« anzeigen und geben Sie dieses hier an:

b) Beschreiben Sie, wie die Ausgabe von Daten in der Konsole dargestellt wird. L

Kapitel 4

Umsetzung des ER-Modells in eine relationale Datenbank

Bei der Überführung eines ER-Modells in eine relationale Datenbank müssen entsprechende Tabellen aufgebaut werden. Dabei gilt die Grundformel, dass sich aus jedem Entitätstyp und jeder Beziehung auch eine eigene Tabelle ergibt. Aus mehreren Gründen wird von dieser Grundformel aber bei den Beziehungen abgewichen.

4.1 Anlegen der Tabellen für Entitäten

Der SQL-Befehl **`CREATE TABLE`** `<tabellenname> (<attribute>);` erstellt eine neue Tabelle in der Datenbank. Die Attribute werden dabei durch Kommata getrennt und das Schlüsselattribut mit **`PRIMARY KEY`** gekennzeichnet. Zusätzlich muss man für jedes Attribut den Typ in der Datenbank auswählen. Die wichtigsten Typen sind folgende:

- **`INT`** eine Ganzzahl.
- **`DOUBLE`** eine Gleitkommazahl.
- **`VARCHAR`** eine Zeichenkette, deren maximale Länge man angeben muss.
- **`TEXT`** ein Textblock in dem auch Umbrüche möglich sind.
- **`TIME`** eine Uhrzeit.
- **`DATE`** ein Datum.

Für den Entitätstyp „Produkt“ aus dem Beispiel ER-Modell würde dann die Erzeugung der Tabelle so aussehen:

```
CREATE TABLE produkt (
  bestellnr INT,
  name VARCHAR(50),
  preis DOUBLE,
  PRIMARY KEY (bestellnr)
);
```

Aufgaben 4.1

a) Erstellen Sie auf Grundlage ihres ER-Modells die nötigen Tabellen für alle Entitätstypen in ihrer eignen Datenbank.

4.2 Umsetzung der Beziehungen

Immer dann, wenn bei einer Beziehung, wie »gehoert zu« aus dem Beispiel, für beide Seiten mehrere Möglichkeiten bestehen, wird eine Tabelle benötigt die mindestens zwei Spalten für die Schlüssel der beteiligten Entitätstypen hat. Zusätzlich kann eine solche Beziehung noch eigene Attribute haben. In dem Beispiel könnte man so eintragen, wann die Zuordnung zur Kategorie getroffen wurde.

Hat man eine Beziehung, bei der einer Entität maximal eine andere zugeordnet wird, so kann man auf diese zusätzliche Tabelle verzichten und direkt in der einen Tabelle ein Attribut für das Schlüsselattribut der anderen Tabelle eintragen. Für die Beziehung »produziert von« aus dem Beispiel bedeutet dieses, dass die Tabelle »Produkt« eine weitere Spalte »hersteller« bekommt, in der immer der eindeutige Name des Herstellers eingetragen wird. Das Relationenschema sieht für Produkt so aus:

```
Produkt(bestellnr, name, preis, ↑hersteller)
```

Der Pfeil vor dem Attribut »hersteller« deutet an, dass es sich hierbei um einen Fremdschlüssel handelt, der eine Beziehung zu einer anderen Tabelle aufbaut.

Eine zusätzliche Spalte kann man mit dem SQL-Befehl `ALTER TABLE <tabellenname> ADD <attributname> <attributtyp>;` an das Ende der Tabelle hinzufügen. Für das obere Beispiel wäre es: `ALTER TABLE produkt ADD hersteller VARCHAR(100);`.

Aufgaben 4.2

a) Geben Sie das Relationenschema für die Beziehung »gehört zu« und den Entitätstyp »kategorie« aus dem Beispiel an. L

b) Ergänzen Sie ihre Datenbank so, dass auch alle Beziehungen aus dem ER-Modell darin abgebildet werden können.

Kapitel 5

Einfügen, Ändern und Löschen von Daten

Das Einfügen in Tabellen erfolgt mit dem **INSERT INTO** Befehl, mit dem auch gleichzeitig mehrere Datensätze eingefügt werden können:

```
INSERT INTO <tabelle> (<spaltennamen>)VALUES (<werte>),(<werte2>);
```

Hierbei müssen nicht immer alle Spalten der Tabelle angegeben werden. Es ist auch möglich Datensätze anzulegen, bei denen nicht für alle Spalten Werte vorliegen. Innerhalb eines Befehls müssen aber zu jeder angegebenen Spalte auch immer ein entsprechender Wert angegeben sein, wie hier im Beispiel:

```
INSERT INTO lehrer (name, vorname, kuerzel) VALUES
  ('Kersten', 'Annegret', 'KA'),
  ('Pichel', 'Manfred', 'PI'),
  ('Vohken', 'Tobias', 'VO');
```

Aufgaben 5.1

a) Fügen Sie die nötigen Datensätze in ihre Datenbank ein, damit diese die Problembeschreibung wiedergibt.

b) Bei Fehleingaben oder Änderungen müssen auch die Datensätze korrigiert werden. Erarbeiten Sie den Aufbau des SQL-Befehls **UPDATE**. L

c) Auch das Löschen muss eine Datenbank ermöglichen. Verfahren Sie beim SQL-

Befehl **DELETE** genauso wie beim **UPDATE**. L

d) Fügen Sie weitere Datensätze ihrer Tabellen hinzu, um eine Grundlage für die weitere Arbeit zu haben.

Kapitel 6

Auswahl von Datensätzen

Der Befehl für die Auswahl und Anzeige von Datensätzen ist der **SELECT**-Befehl, der bereits in 3.1 kurz vorgestellt wurde. Dabei wurde aber nur die einfachste Möglichkeit gezeigt. Ihre Stärken spielt eine relationale Datenbank erst dann aus, wenn verschiedene Dinge miteinander kombiniert werden. Im ersten Schritt kann dieses durch die Kombination und Verneinung von Bedingungen über die Wörter **AND**, **OR** und **NOT** geschehen.

Aufgaben 6.1

a) Erstellen Sie eine Abfrage, die ihnen alle Schüler ihrer eigenen Datenbank liefert, die nicht in der Stufe Q1 sind und geben Sie diese hier an. Überprüfen Sie diese Anfrage auch im System. L

b) Es sollen nur alle Kurse ausgegeben werden, die von Herrn Hospital oder Frau Umardo unterrichtet werden. Geben Sie eine entsprechende SQL-Anweisung hier an und überprüfen Sie diese. L

6.1 Vergleichsoperatoren

In den bisherigen Abfragen wurden nur direkte Vergleiche mit dem Gleichheitszeichen durchgeführt. Bei Zahlenwerten können aber auch Größenvergleiche durchgeführt werden. Dazu dienen die aus anderen Bereichen bekannten Vergleichsoperatoren `>` `<`, `>=` und `<=`. Um auf Ungleichheit zu testen kann sowohl `<>` als auch `!=` genutzt werden.

Durch die Verknüpfung mit einem **AND** kann man so Elemente in einem Bereich auswählen. Dafür bietet MySQL die einfache Möglichkeit eines **BETWEEN** `<min>` **AND** `<max>` an. Ähnlich sieht es aus, wenn man verschiedene Werte für ein Attribut zutreffen dürfen. Diese müssen nicht mit **OR** verknüpft werden, sondern können als Aufzählung **IN** `(<wert1>, <wert2>, <wert3>)` aufgeführt werden. Alle Werte sind dabei durch Komma separiert.

Ist ein Attributwert nicht gesetzt, so kann dieses mit Hilfe von **IS NULL** überprüft werden. Bei Zeichenketten bietet der Vergleich mit **LIKE** auch weitere Unterscheidungsmöglichkeiten als das Gleichheitszeichen. In dem angegeben Vergleichswert können zusätzlich auch `%` und `_` angegeben werden. Dabei steht das Prozentzeichen für beliebig viele Zeichen, während der Unterstrich genau für ein einzelnes Zeichen steht. So liefert eine Suche mit `name` **LIKE** `'B%n'` auch Namen wie Benjamin, Ben aber Bn. Bei `name` **LIKE** `'B_n'` würde nur Ben aus den vorherigen Beispielen in der Ergebnismenge vorkommen. Diese Steuerzeichen können innerhalb des Suchwertes auch mehrfach vorkommen und miteinander kombiniert werden. Sucht man aber das Prozentzeichen oder den Unterstrich, so muss man dieses mit `\%` bzw. `\_` machen.

Aufgaben 6.2

a) Geben Sie das Äquivalent der Anfrage `SELECT * FROM mitglieder WHERE eintrittsjahr BETWEEN 1980 AND 2000;` an, dass nicht mit **BETWEEN** funktioniert. L

b) Schreiben Sie die Namen auf, die die Anfrage `SELECT name FROM mitglieder WHERE name LIKE '%a%er';` auf die Datenbank des Sportvereins liefert. Überprüfen Sie dieses anschließend durch das Ausführen der Abfrage. L

6.2 Abfrage über mehrere Tabellen

Durch das Setzen der Beziehungen liegen viele Informationen über mehrere Tabellen verteilt. Daher kann man eine Abfrage auch an mehrere Tabellen gleichzeitig stellen. Dazu müssen nur die Tabellennamen durch Komma getrennt im **FROM**-Teil angegeben werden. Sollten dabei Attribute in den verschiedenen Tabellen die gleichen Namen tragen, so kann man vor den Attributnamen, wie bei `Produkt.name` den Tabellennamen mitangeben.

Aufgaben 6.3

a) Lassen Sie sich gleichzeitig den Inhalt der Tabellen »mitglieder« und »tarif« aus der Datenbank »sportverein« anzeigen. Erläutern Sie das Ergebnis ihre Anfrage. L

b) Verändern Sie durch Bedingungen ihre obige Anfrage, so dass sie sinnvolle Ergebnisse erhalten. Schreiben Sie ihre Anfrage hier auf: L

6.3 Kartesisches Produkt

Wird eine Abfrage über zwei Tabellen durchgeführt, so wird jede Zeile der ersten Tabelle mit jeder Zeile der zweiten Zeile verknüpft. Dieses nennt man das kartesische Produkt. Durch eine Bedingung, bei der zwei Attribute aus den beiden Tabellen übereinstimmen müssen, kann man dafür sorgen, dass nur solche Zeilen ausgewählt werden, bei denen es auch eine Übereinstimmung gibt. So würde man bei dem Verknüpfen von Mitgliedern und Tarifen die Bedingung `mitglieder.zahlt = tarif.name` setzen.

Da bei einigen Fällen die Abfragen sehr groß und unübersichtlich werden können, besonders wenn mehrere Bedingungen zutreffen müssen, gibt es die Möglichkeit Abkürzungen einzufügen. Mit dem Wort **AS** kann man sowohl Tabellennamen als auch Attribute abkürzen oder auch umbenennen. So könnte die obige Abfrage auch so aussehen: `SELECT * FROM mitglieder AS m, tarif AS t WHERE m.zahlt = t.name;`.

Aufgaben 6.4

a) Entwickeln Sie eine Abfrage, die ihnen zu jedem Schüler aus ihrer Datenbank den Stufenbegleiter angibt. Dabei sollen nur die Namen des Schülers und des Lehrers angegeben werden. Nutzen Sie dazu entsprechende Abkürzungen um die Attribute deutlich voneinander unterscheiden zu können und geben Sie ihre Anfrage hier an: L

6.4 Join

Eine äquivalente Abfrage zu der Verknüpfung der beiden Tabellen »mitglieder« und »tarif« kann auch mit Hilfe von **JOIN** erzielt werden: `SELECT * FROM mitglieder JOIN tarif ON mitglieder.zahlt = tarif.name;`. Dieses ist deshalb interessant, weil der Join um weitere Eigenschaften erweitert werden kann. So liefert die Anfrage `SELECT * FROM mitglieder LEFT JOIN vorstand ON mitglieder.id = vorstand.id;` alle Mitglieder und gibt an, wenn sie einen Vorstandsposten haben. Bei allen Mitgliedern, die keinen Vorstandsposten haben wird der Wert für den Posten in der Ergebnistabelle auf NULL gesetzt.

Bei einer solchen Abfrage spricht man von einem Left-Join. Analog dazu gibt es auch einen Right-Join, bei dem auch alle Vorstandsposten aufgezählt würden, die nicht besetzt sind.

Aufgaben 6.5

a) Vollziehen Sie die oben angegeben Abfragen auf der Datenbank nach.

b) Lassen Sie sich alle Mitglieder mit den Sportarten ausgeben und schreiben Sie diese Anfrage hier auf: L

c) Entwickeln Sie Anfragen, mit denen Sie nur die Namen der aktiven bzw. passiven Mitglieder ausgeben. Passive Mitglieder sind solche, die an keiner Sportart teilnehmen. Geben Sie die Anfragen auf und geben Sie Schönheitsfehler in den Ergebnissen mit an. Hinweis: Bedingungen bzgl. NULL müssen **IS NULL** oder **IS NOT NULL** lauten. L

6.5 Projektion

Bei der letzten Aufgabe trifft man auf ein nicht befriedigendes Ergebnis: Bei den aktiven Mitgliedern kommen einige Namen doppelt vor. An dieser Stelle hilft die Projektion.

Bei der Projektion werden Zeilen in der Ergebnistabelle, die in allen Attributen übereinstimmen, auf eine einzelne Spalte projiziert. Dieses lässt sich durch das Schlüsselwort **`DISTINCT`** direkt hinter dem **`SELECT`** erreichen. So taucht im Beispiel der aktiven Mitglieder jeder Name, bzw. die Kombination aus id und Name nur einmal auf, je nachdem, was in der Anfrage mit ausgegeben werden sollte.

Aufgaben 6.6

a) Verbessern Sie ihre Anfrage aus der letzten Aufgabenstellung.

b) Erläutern Sie, weshalb die Projektion bei einem **`SELECT DISTINCT * FROM`** bei der Ausgabe der aktiven Mitglieder scheinbar keine Wirkung zeigt. L

Kapitel 7

Änderungen an Tabellen

Im Abschnitt 4.2 wurde bereits deutlich, dass beim Anlegen der Tabellen in der Datenbank nicht immer alle Punkte richtig beachtet wurden. In solchen Fällen kann es nötig werden, Tabellen zu entfernen oder ihre Struktur zu verändern. Dieses kann das bereits vorgestellte Hinzufügen von Attributen sein. Es ist aber auch möglich, Attribute zu entfernen, ihren Typ zu ändern oder auch den Schlüssel einer Tabelle neu zu setzen. Dabei ist aber Vorsicht geboten. Gerade bei großen Datenbanken können solche Änderungen zu Problemen führen und dauern in der Regel sehr lange.

7.1 Tabellen löschen, leeren oder umbenennen

Das Löschen von Tabellen geschieht einfach mit dem Befehlt **DROP TABLE** <tabelle>;. Damit werden alle Daten und die Struktur der Tabelle gelöscht. Will man hingegen nur eine Tabelle leeren, so kann man **TRUNCATE TABLE** <tabelle>; nutzen, was auch den Vorteil hat, dass ein möglicher Zähler für eine eindeutige ID wieder zurückgesetzt wird. Mit **RENAME TABLE** <altername> **TO** <neuername>; erfolgt ein mögliches Umbenennen einer Tabelle.

7.2 Struktur der Tabellen bearbeiten

Mit **ALTER TABLE** <tabelle> **ADD** <attributname> <attributtyp>; wird, wie bereits geschrieben eine neues Attribut hinzugefügt. Dieses wird immer an das Ende der Relation an gehangen. Es lässt sich aber auch ein **FIRST** oder **AFTER** <attributname> an den Befehl anhängen, um so die Position zu beeinflussen. Das Entfernen eines Attributs geschieht mit **ALTER TABLE** <tabelle> **DROP COLUMN** <attributname>;.

Sowohl das Umbenennen eines Attributs als auch das Ändern seines Typs geschieht mit Hilfe des Befehls **ALTER TABLE** <tabelle> **CHANGE** <altername> <neuername> <attributtyp>;. Soll also nur der Typ geändert werden, so müssen der alte und der neue Name identisch

sein. Analog muss bei einer Umbenennung immer der bisherige Typ mit angegeben werden.

7.3 Schlüssel ändern

Soll im Nachhinein ein Schlüssel für eine Tabelle gesetzt werden, so geschieht dieses mit `ALTER TABLE <tabelle> ADD PRIMARY KEY (<attribut>);`. Sollen zwei oder mehrere Attribute gemeinsam den Schlüssel bilden, so können sie innerhalb der Klammen durch Komma getrennt aufgeführt werden. Will man den Schlüssel auf ein anders Attribut ändern, so muss dieser erst mit `ALTER TABLE <tabelle> DROP PRIMARY KEY;` von der Tabelle entfernt werden, bevor er anschließend neu gesetzt wird.

Kapitel 8

Normalisierung

Bei der Arbeit mit Datenbanken besteht das Ziel darin, möglichst einfach Daten miteinander verknüpfen und einzelne Daten abfragen zu können. Um dieses zu ermöglichen und überprüfbar zu machen, werden Datenbanken mittels der Normalisierung in eine Normalform überführt. Je nachdem, wie weit man diese Normalisierung durchführt, werden unterschiedliche aufeinander aufbauende Normalformen erreicht. Hier betrachten wir nur die drei ersten Normalformen.

8.1 1. Normalform

Eine Datenbank liegt dann in der ersten Normalform vor, wenn folgendes auf sie zutrifft:

Die Datenbank enthält nur atomare Attribute.

Dieses bedeutet, dass nur einfache und nicht zusammengesetzte Attribute erlaubt sind. Ein klassisches Beispiel ist die Adresse, die sich aus den einzelnen Attributen Straße, Postleitzahl und Ort zusammensetzt und daher für die erste Normalform in unterschiedlichen Attributen erfasst werden muss.

Eine weitere Möglichkeit der Verletzung der ersten Normalform liegt in einer Aufzählung. Ein solcher Fall liegt in der folgenden Tabelle bei den Herstellungsangaben vor, in die angelieferten Produkte eines Ladens aufgelistet sind:

bestellnr	name	hersteller	ort	herstellungsangaben
834949	Gulaschsuppe	Magnat	Datteln	39232988: 14.05.13, 39232993: 15.05.13
834154	Erbsensuppe	Magnat	Datteln	39234540: 04.08.13, 39234546: 05.08.13
154648	Bohnen	B & B	Zürich	45421544: 01.06.13

Die Herstellungsangaben beinhalten dabei die Chargennummer und das Herstellungsdatum. Die Chargennummer bezeichnet dabei die laufende Nummer, die vergeben wird, wenn in einem Topf eine neue Portion angerührt wird, aus dem dann in die Einzelverpackungen abgefüllt wird. Um diese Tabelle in die erste Normalform zu bringen, muss für

jede Chargennummer eine neue Zeile eingeführt werden. Dabei wird die Chargennummer zu einem neuen Schlüssel, da die Bestellnummer nicht mehr eindeutig ist.

bestellnr	name	hersteller	ort	chargennummer	datum
834949	Gulaschsuppe	Magnat	Datteln	39232988	14.05.13
834949	Gulaschsuppe	Magnat	Datteln	39232993	15.05.13
834154	Erbsensuppe	Magnat	Datteln	39234540	04.08.13
834154	Erbsensuppe	Magnat	Datteln	39234546	05.08.13
154648	Bohnen	B & B	Zürich	45421544	01.06.13

Aufgaben 8.1

a) Begründen Sie, weshalb die Datenbank »sportverein« nicht in der ersten Normalform vorliegt. Geben Sie außerdem die Befehle an, mit der die Struktur der Datenbank entsprechend geändert wird.

b) Zählen Sie Vorteile auf, die sich durch die Umformung in die erste Normalform ergeben.

c) Überführen Sie ihre Datenbank für das König-Joseph-Gymnasium in die erste Normalform.

8.2 2. Normalform

Eine Datenbank, die die obere Tabelle mit den angelieferten Waren enthält, ist nicht in der zweiten Normalform. Für sie gilt nicht die folgende Bedingung:

> Die Datenbank entspricht der 1. Normalform und jedes nicht Schlüsselattribut hängt vom ganzen Schlüssel der jeweiligen Tabelle ab (Wenn die Tabelle zwei Schlüssel enthält, müssen alle Attribute auch von beiden Schlüsseln abhängig sein).

Die in der Bedingung angesprochene Abhängigkeit hat folgende Bedeutung: Wenn ein Attribut A von einem anderen Attribut B abhängig ist, dann ist bei einem gleichbleibenden Wert für das Attribut B der Wert für das Attribut A immer gleich.

In dem Beispiel ist die Bedingung für die zweite Normalform durch die Spalten »name«, »hersteller« und »ort« verletzt. Dieses lässt sich daran erkennen, dass eine Änderung der Chargennummer keine Änderung in diesen Attributen nötig macht. Daher wird die einzelne Tabelle in zwei Tabellen aufgespalten:

bestellnr	name	hersteller	ort
834949	Gulaschsuppe	Magnat	Datteln
834154	Erbsensuppe	Magnat	Datteln
154648	Bohnen	B & B	Zürich

bestellnr	chargennummer	datum
834949	39232988	14.05.13
834949	39232993	15.05.13
834154	39234540	04.08.13
834154	39234546	05.08.13
154648	45421544	01.06.13

Aufgaben 8.2

a) Bei dieser Normalform geht es besonders darum, doppelte Speicherung von Informationen zu vermeiden. Erläutern Sie die Vorteile dieser Reduzierung.

b) Sehr wahrscheinlich wird sich ihre Datenbank des König-Joseph-Gymnasiums bereits in der zweiten Normalform befinden, nachdem Sie die Regeln für die erste Normalform darauf angewandt haben. Überprüfen Sie, ob Ihre Datenbank in der zweiten Normalform vorliegt. Wenn ja, geben Sie an, an welcher Stelle Sie bereits dafür gesorgt haben.

c) Wenn Ihre Datenbank noch nicht in der zweiten Normalform vorliegt, ändern Sie diese entsprechend.

8.3 3. Normalform

Oft gibt es diese Mehrfachspeicherung von Daten nicht nur in Abhängigkeit von Schlüsselattributen, wie sie mit der zweiten Normalform aufgelöst wurden, sondern auch in Abhängigkeit von anderen Attributen. Ein bekanntes Beispiel ist hier eine Postleitzahl, die in Deutschland in den meisten Fällen eindeutig einer Stadt zugeordnet ist. Um

diese Mehrfachspeicherung aufzulösen wird eine Datenbank in die dritte Normalform gebracht:

> Die Datenbank entspricht der 2. Normalform und es gibt keine Abhängigkeiten zwischen nicht Schlüsselattributen (Ein Attribut hängt nicht von einem anderen nicht Schlüsselattribut ab).

Das oben genannte Beispiel befindet sich auch nicht in der dritten Normalform. Die Herstellungsadresse und damit hier der Ort ist vom Hersteller abhängig. Deshalb müsste dieser aus der ersten Tabelle heraus gelöst werden:

bestellnr	name	hersteller
834949	Gulaschsuppe	Magnat
834154	Erbsensuppe	Magnat
154648	Bohnen	B & B

bestellnr	chargennummer	datum
834949	39232988	14.05.13
834949	39232993	15.05.13
834154	39234540	04.08.13
834154	39234546	05.08.13
154648	45421544	01.06.13

hersteller	ort
Magnat	Datteln
B & B	Zürich

Aufgaben 8.3

a) Im oberen Beispiel wird davon ausgegangen, dass jeder Hersteller nur an einem Ort produziert. Äußern Sie sich bezüglich der Bedingung für die dritte Normalform, wenn dieses nicht der Fall ist.

b) Überprüfen Sie, ob ihre Datenbank in der dritten Normalform vorliegt und ändern Sie diese sonst entsprechend.

8.4 Grenzen der Normalisierung

Besonders aus Effizienzgründen bei Abfragen kommt es vor, dass Datenbanken nicht bis zur dritten Form normalisiert werden. So kann es ein großer Geschwindigkeitsvorteil sein, wenn bei einer großen Anzahl von Abfragen nicht erst mehrere riesige Tabellen miteinander verknüpft werden müssen. Damit einher geht aber, dass Daten mehrfach vorliegen und es dabei auch zu Inkonsistenzen kommen kann. Diese liegen dann vor, wenn an verschiedenen Stellen die eigentlich gleichen Daten unterschiedlich sind.

Arbeitet man mit nicht normalisierten Datenbanken, so muss man in den Programmen, die auf der Datenbank agieren, durch entsprechende Strukturen dafür Sorge tragen, dass alle Daten konsistent sind. Dieses gilt besonders für das Ändern von Einträgen. Aber auch beim Löschen muss darauf geachtet werden, dass alle Einträge korrekt und entfernt werden.

Kapitel 9

Einbinden in Java Programme

Java bietet die Möglichkeiten, innerhalb eines Programms Anfragen an eine MySQL-Datenbank zu stellen. Innerhalb des Pakets java.sql befinden sich die nötigen Klassen um allgemein mit SQL zu arbeiten. Zusätzlich wird noch der für MySQL passende Treiber »org.gjt.mm.mysql.Driver« benötigt, der die Verbindung zur MySQL-Datenbank herstellt. Dieser ist in der Regel nicht bei Java mitgeliefert, sondern muss nachinstalliert werden.

Mit einem kleinen Programm lässt sich testen, ob dieser Treiber richtig installiert ist und sich auch laden lässt. Dabei sollten die Hinweise aus dem Anhang B beachtet werden.

```
1  import java.sql.*;
2
3  public class LoadDriver {
4    public static void main(String[] args) {
5      try {
6        System.out.println("* Treiber laden");
7        Class.forName("org.gjt.mm.mysql.Driver").newInstance();
8      }
9      catch (Exception e) {
10       System.err.println("Laden des Treibers ist fehlgeschlagen.");
11       e.printStackTrace();
12     }
13   }
14 }
```

Das Laden des Treibers geschieht hier in einer try-catch-Umgebung, die dafür sorgt, dass bei auftretenden Fehlern das Java-Programm nicht komplett beendet wird, sondern auf die Fehler reagiert werden kann. Dieses sollte bei allen MySQL-Befehlen durchgeführt werden, weil nicht zu jeder Zeit sichergestellt ist, ob die Verbindung zur Datenbank aufgebaut werden kann und ob z. B. die Datenbankstruktur mit der Abfrage übereinstimmt.

Aufgaben 9.1

a) Überprüfen Sie mit dem oben angegebenen Programm die Korrektheit der Treiberinstallation auf ihrem System.

9.1 Verbindung aufbauen

Nach dem Laden des Treibers muss eine Verbindung mit der Datenbank aufgebaut werden. Mit folgenden Zeilen, die auch wieder in einem try-catch-Block stehen sollten, bekommt man ein Connection-Objekt:

```
1 String url = "jdbc:mysql://"+hostname+":"+port+"/"+dbname;
2 Connection conn = DriverManager.getConnection(url, user,
  password);
3 Statement stmt = conn.createStatement();
```

Mit Hilfe dieses Connection-Objekts kann man, wie in der dritten Zeile angegeben, ein Statement-Objekt bekommen. Dieses ist nötig um Abfragen und Anweisungen an die Datenbank zu senden. Dabei ist zu beachten, dass beim oberen Beispiel die Variablen für z. B. den Hostname zuvor entsprechend gesetzt sein müssen.

9.2 Abfrage absetzen und auswerten

Mit dem Statement-Objekt, lassen sich die MySQL-Anweisungen auf der Datenbank ausführen. Dazu übergibt man die Anweisung der Methode `executeQuery(String Query)` als Zeichenkette, wie im folgenden Beispiel:

```
1 String sqlCommand = "SELECT␣*␣FROM␣mitglieder;";
2 ResultSet rs = stmt.executeQuery(sqlCommand);
```

Die Antwort auf die Anfrage wird in einem ResultSet-Objekt zurückgegeben. Liefert die Anfrage eine Tabelle, so kann diese mithilfe einer internen Referenz wie eine Liste durchwandert werden. Zu Beginn steht diese Referenz immer vor der ersten Zeile und kann mit den Methoden `next()` und `previous()` vor und zurück bewegt werden.

Wie im folgenden Beispiel zu erkennen ist, kann man mit verschiedenen get-Methoden auf die Elemente in den gewünschten Typen zugreifen. Dabei kann die Angabe der Spalte sowohl über den Namen, als auch ihre Nummer geschehen. Dabei ist zu beachten, dass die Spalten von eins an durchnummeriert sind. So werden nacheinander alle Zeilen des Resultats durchlaufen.

```
while (rs.next()) {
  int id = rs.getInt(1);
  String name = rs.getString(2);
  String anschrift = rs.getString("anschrift");
  System.out.println(id + "␣" + name + "␣" + anschrift);
}
```

Aufgaben 9.2

a) Erstellen Sie ein Java-Programm, mit dem Sie zuerst immer die Betreuungslehrer und andere Informationen über eine Stufe ausgegeben werden. Daran anschließend sollen die Schüler einer jeden Stufe ausgeben werden.

9.3 Beispielprogramm

Ein komplettes Java-Programm, dass alle beschriebenen Element beinhaltet sieht dann wie folgt aus:

```
import java.sql.*;

public class SimpleDbExample {

  public static void main(String[] args) {
    // Diese Eintraege werden zum
    // Verbindungsaufbau benoetigt.
    final String hostname = "localhost";
    final String port = "3306";
    final String dbname = "sportverein";
    final String user = "javatest";
    final String password = "test";

    Connection conn = null;

    try {
      System.out.println("*␣Treiber␣laden");
      Class.forName("org.gjt.mm.mysql.Driver").newInstance();
    }
    catch (ReflectiveOperationException e) {
      System.err.println("Laden␣des␣Treibers␣ist␣fehlgeschlagen.");
      e.printStackTrace();
    }

    try {
      System.out.println("*␣Verbindung␣aufbauen");
```

```
27        String url = "jdbc:mysql://"+hostname+":"+port+"/"+dbname;
28        conn = DriverManager.getConnection(url, user, password);
29
30        System.out.println("* Statement beginnen");
31        Statement stmt = conn.createStatement();
32
33        String sqlCommand = "SELECT * FROM mitglieder;";
34        ResultSet rs = stmt.executeQuery(sqlCommand);
35
36        System.out.println("* Ergebnisse anzeigen");
37        while (rs.next()) {
38          int id = rs.getInt(1);
39          String name = rs.getString(2);
40          String anschrift = rs.getString("anschrift");
41          System.out.println(id + " " + name + " " + anschrift);
42        }
43
44        System.out.println("* Datenbank-Verbindung beenden");
45        conn.close();
46      }
47      catch (SQLException sqle) {
48        System.out.println("SQLException: " + sqle.getMessage());
49        System.out.println("SQLState: " + sqle.getSQLState());
50        System.out.println("VendorError: " + sqle.getErrorCode());
51        sqle.printStackTrace();
52      }
53    } // ende: public static void main()
54
55 } // ende: public class SimpleDbExample
```

Kapitel 10

Weitere Abfragen

Die in Abschnitt 6 gezeigten Möglichkeiten für Datenbankanfragen stellen nur einen kleinen Ausschnitt der Möglichkeiten dar. Besondere Stärke spielt SQL dann aus, wenn man Abfragen miteinander kombiniert oder auch eingebaute Funktionen nutzt, mit denen man Ergebnisse manipulieren oder ausrechnen lassen kann. Diese Möglichkeiten werden in diesem Abschnitt gezeigt.

10.1 Zusammenfassen von Anfragen

Es kommt vor, dass man verschiedene Tabellen in einer Datenbank hat, die eine sehr ähnliche Relation besitzen. Dieses ist zum Beispiel bei den Lehrern und den Schülern der Fall. Gibt es Fälle, in denen man auf beide gleichzeitig zugreifen will, so muss man die Ergebnisse der Anfragen zusammensetzen. Dieses könnte der Fall sein, wenn man eine Gesamtliste der Schulgemeinschaft erhalten will. In MySQL gelingt dieses Zusammenfügen mit dem Schlüsselwort **`UNION`**, das zwischen zwei Abfragen gesetzt werden muss wie hier: `SElECT name FROM lehrer UNION SELECT name FROM schueler;`. Wichtig ist dabei, dass beide Anfragen die gleiche Anzahl an Attributen besitzen und auch die Reihenfolge übereinstimmt.

Aufgaben 10.1

a) Lassen Sie sich alle Mitglieder und Trainer des Vereins gemeinsam angeben. Schreiben Sie die dazu nötige Abfrage hier auf und geben Sie an, was mit Personen passiert, die gleichzeitig Mitglied und als Trainer tätig sind. L

10.2 Anfragen auf Ergebnisse von Anfragen (Unterabfragen)

Eine Stärke von SQL ist es, dass das Ergebnis einer Anfrage auch direkt wieder in einer nächsten Anweisung eingesetzt werden kann. So kann die Tabelle, die eine Anfrage liefert auch direkt mit einer anderen verknüpft werden oder darauf auch wieder eine andere Anfrage gestartet werden. Das häufigste Beispiel ist aber, dass die Anfrage mehrere Werte liefert, die für eine Bereichsabfrage mit **IN** durchgeführt werden soll. So möchte man z. B. alle Mitglieder des Sportvereins haben, die einen Posten im Vorstand haben. Auf den Join soll verzichtet werden. Daher liefert `SELECT id FROM vorstand` eine Liste mit den ID-Nummern, der Mitglieder die einen solchen Posten haben. Zusammengesetzt zur Abfrage `SELECT name FROM mitglieder WHERE id IN (SELECT id FROM vorstand);` bekommt man nun alle Namen der Vorstandsmitglieder aufgelistet.

Aufgaben 10.2

a) Das Mitglied »Christina Schreiner« will an der Sportart »Schwimmen« teilnehmen. Geben Sie die entsprechende Anweisung an, mit dem dieses in die Datenbank des Sportvereins aufgenommen wird. Verzichten Sie dabei auf direkte Angaben weiterer Werte, wie z. B. der ID. L

10.3 Sortieren

Bei alle Anfragen die an das Datenbanksystem gestellt werden, sind die Elemente in der Antwort in der Reihenfolge aufgeführt, wie sie auch in der Datenbank selber vorliegen. Da dieses aber nicht immer den eigenen Wünschen entspricht, kann man die **SELECT**-Anweisung um ein **ORDER BY** <spaltenname> ergänzen. Dabei wird standardmäßig aufsteigend sortiert. Dieses kann man aber auch mit dem Schlüsselwort **ASC** zusätzlich festlegen. Die Gegenrichtung erfolgt mit **DESC**.

Eine Sortierung alle Mitglieder nach ihrem Eintrittsjahr erfolgt daher mit `SELECT * FROM mitglieder ORDER BY eintrittsjahr ASC;`. Möchte man mehrere Sortierungskriterien anwenden, so können diese wieder mit Komma getrennt hintereinander geschrieben werden. So sortiert `SELECT * FROM mitglieder ORDER BY eintrittsjahr DESC, name ASC` erst absteigend nach dem Eintrittsjahr und dann als zweites nach dem Namen.

10.4 Funktionen

Bei jeder Anfrage, die man an das Datenbanksystem stellt, wird auch immer die Anzahl der betroffen Zeilen mit angegeben. Dafür wird intern die Zählfunktion `COUNT(*)` genutzt, die auch direkt genutzt werden kann. Jede Funktion wird bei einer Abfrage anstelle eines ausgewählten Attributs angegeben. So zählt `SELECT COUNT(*)FROM mitglieder;` die Mitglieder und gibt die Anzahl aus.

Weitere Funktionen sind `SUM(<spalte>)` um die Werte der übergebenen Spalte zu summieren, `MAX(<spalte>)` und `MIN(<spalte>)` um das Maximum oder das Minimum einer Spalte zu finden.

Aufgaben 10.3

a) Bestimmen Sie die Einnahmen, die der Sportverein laut Datenbank durch die Mitgliedsbeiträge erhält. L

10.5 Arithmetik

Neben dem Ausführen von Funktionen kann man innerhalb von Abfragen auch Rechnungen mit `+`,`-`,`*` und `/`. Auch lassen sich Klammern setzen, um die Reihenfolge zu beeinflussen. So liefert die Abfrage `SELECT SUM(preis)/ COUNT(*)FROM tarif;` den durchschnittlichen Preis aller Tarife des Sportvereins. Gleiches würde aber auch die Funktion `AVG(<spalte>)` liefern.

10.6 Gruppieren

Mit der Möglichkeit durch `GROUP BY <spalte>` innerhalb einer Abfrage Gruppen zu erstellen, werden besonders die Möglichkeiten der eingebauten Funktionen erweitert. Dabei werden Zeilen bei denen das zu gruppierende Element gleich ist zu einer Gruppe zusammengestellt und nur die jeweils erste Zeile ausgegeben. Es lassen sich aber auch die Elemente einer Gruppe zählen und so gibt `SELECT name, COUNT(*)FROM teilnehmen GROUP BY name;` zu jeder Sportart an, wie viele Mitglieder daran teilnehmen.

Kapitel 11

Sicherheit

Es fehlen noch etwas zum Stichwort SQL-Injection.

Kapitel 12

Nachwort

Im Vorwort (1) haben Sie zu verschiedenen Punkten Stellung genommen. Äußern Sie sich erneut zu den Punkten und vergleichen Sie diese anschließend mit ihren Angaben aus dem Vorwort.

Kapitel 13

Sammlung von Aufgaben

Zur Übung und Vertiefung sind hier verschiedene Aufgaben angegeben.

Aufgaben 13.1

a) Lassen Sie sich zu jedem Mitgliedsnamen des Sportvereins angeben, an wie vielen Sportarten die Mitglieder jeweils teilnehmen und geben Sie ihre Anfrage hier an.
L

Weitere Aufgaben zur Vertiefung benötigt

Anhang A

Installation von MySQL

Die Installation eines MySQL-Servers ist auf allen drei gebräuchlichen Plattformen (Linux, Windows, Mac) möglich. Soll zusätzlich auch phpMyAdmin genutzt werden, muss zum MySQL-Server auch ein Webserver mit PHP Unterstützung installiert werden. Dazu gibt es bereits viele Anleitungen im Internet, auf die an dieser Stelle verwiesen wird:

A.1 Debian/Ubuntu

Unter Debian/Ubuntu müssen nur die Pakete mysql-server, mysql-client und phpmyadmin mit den zugehörigen Abhängigkeiten installiert werden. Dann steht eine entsprechende Grundinstallation zur Verfügung. Weitergehende Hilfe findet man auch auf den Seiten `http://wiki.ubuntuusers.de/MySQL` und `http://wiki.ubuntuusers.de/MySQL/Werkzeuge`.

A.2 Windows

Für die Installation von MySQL unter Windows gibt es passende Quellen unter `http://www.mysql.de/why-mysql/windows/`. Es bietet sich aber auch an XAMPP zu installieren. Dann hat man ein Komplettsystem aus dem Apachewebserver, PHP und MySQL. Auch phpMyAdmin ist direkt mit installiert. Zu finden ist XAMPP auf `http://www.apachefriends.org/de/index.html`.

Anhang B

Installieren des MySQL-Treibers für Java

B.1 Debian/Ubuntu

Der benötigte Treiber kann unter Debian bzw. Ubuntu mit dem Paket libmysql-java installiert werden. Er befindet sich dann in einer .jar-Datei im Verzeichnis `/usr/share/java/`. Je nach Installation kann es dann vorkommen, dass beim Ausführen des Java-Programms der Classpath explizit mit angegeben werden muss. Ein Aufruf würde dann so aussehen:
`java -cp /usr/share/java/*:. SimpleDbExample`

Es ist auch möglich, den Classpath in der Enviroment fest mit anzugeben. Dazu fügt man der Datei `.profile` im Homeverzeichnis die Zeile **`export`** `CLASSPATH="/usr/share/java/*:."` hinzu bzw. ändert sie so ab, dass sie `/usr/share/java/*` enthält. Will man dieses für alle Benutzer machen, so sollte dieses in die Datei `/etc/enviroment` übernommen werden.

Anhang C

Lösungen

3.1 a `sportverein (id, name, anschrift, eintrittsjahr, geschlecht, ↑zahlt)`
`sportart (name, bemerkung)`
Zurück

3.1 b `SELECT * FROM mitglieder LIMIT 0 , 30`
Er wählt so alle Attribute der Tabelle mitglieder aus. Es werden aber nur die ersten dreißig dargestellt. Zurück

3.2 b Es wird mit entsprechenden ASCII-Zeichen eine Tabelle erstellt und diese mit den Inhalten gefüllt. Die Spaltenköpfe enthalten jeweils die Attributnamen Zurück

4.2 a `gehoertzu (↑produkt, ↑kategorie)`
`kategorie (name, ↑unterkategorie)`
Zurück

5.1 b `UPDATE <tabelle> SET <attribut> = <attributwert> WHERE <bedingung>;`
Man gibt an, ich welcher Tabelle es Änderungen geben soll. Dann muss man die Attribute mit ihren neuen Attributwerten angeben. Sind es mehrere, so müssen sie durch Kommata getrennt werden. Anschließend muss man noch die Bedingung angeben, bei welchen Datensätzen diese Änderung überhaupt vorgenommen werden soll. Zurück

5.1 c `DELETE FROM <tabelle> WHERE <bedingung>;`
Hier wird die Tabelle angegeben, in der Datensätze gelöschten werden soll. Welche dieses sind, wird durch die Bedingung festgelegt. Zurück

6.1 a `SELECT * FROM schueler WHERE stufe NOT LIKE 'Q1';` oder `SELECT * FROM schueler WHERE stufe != 'Q1';` Zurück

6.1 b `SELECT * FROM kurse WHERE lehrer = 'Hospital'OR lehrer = 'Umardo';` Zurück

6.2 a `SELECT * FROM mitglieder WHERE eintrittsjahr >= 1980 AND eintragen <= 2000;` Zurück

6.2 b Franziska Weber, Christina Schreiner, Alexander Beyer, Tobias Zimmer Zurück

6.3 a `SELECT * FROM mitglieder, tarif;` Nun wird jeder Datensatz von Mitglieder mit jedem Datensatz von tarif kombiniert. Das Ergebnis nennt man auch ein Kartesisches Produkt. Zurück

6.3 b `SELECT * FROM mitglieder, tarif WHERE mitglieder.zahlt = tarif.name;` Zurück

6.4 a `SELECT s.name AS schuelername, kurs.lehrer AS lehrername FORM schueler AS s, stufe WHERE schueler.stufe = stufe.name;` Zurück

6.5 b `SELECT * FROM mitglieder LEFT JOIN teilnehmen ON mitglieder.id = teilnehmen.id;` Zurück

6.5 c Aktive Mitglieder: `SELECT * FROM mitglieder LEFT JOIN teilnehmen ON mitglieder.id = teilnehmen.id WHERE teilnehmen.name IS NOT NULL;`
Passive Mitglieder: `SELECT * FROM mitglieder LEFT JOIN teilnehmen ON mitglieder.id = teilnehmen.id WHERE teilnehmen.name IS NULL;` Zurück

6.6 b Ein `DISTINCT` fasst nur solche Zeilen zusammen, die in allen Attributen übereinstimmen. Zurück

10.1 a `SELECT name, anschrift, geschlecht FROM mitglieder UNION SELECT name, anschrift, geschlecht FROM trainer;` Zurück

10.2 a `INSERT INTO teilnehmen (id, name)VALUES ((SELECT id FROM mitglieder WHERE name = "Christina_Schreiner"),"Schwimmen")` Zurück

10.3 a Die Abfrage `SELECT SUM(tarif.preis)FROM mitglieder JOIN tarif ON mitglieder.zahlt = tarif.name;` liefert 1868. Zurück

13.1 a `SELECT mitglieder.name, COUNT(*)FROM mitglieder JOIN teilnehmen ON mitglieder.id = teilnehmen.id GROUP BY teilnehmen.id;` Zurück

Literaturverzeichnis

[Löffler 2010] Löffler, Susanne: *Von der objektorientierten Modellierung zur Datenbank – ein Konzept und seine Umsetzung mit Mobiltelefonen in der gymnasialen Oberstufe*. Hamm, Studienseminar für Lehrämter an Schulen – Seminar für das Lehramt für Gymnasien/Gesamtschulen, Hausarbeit gemäß OVP, Februar 2010. – URL http://www.ham.nw.schule.de/pub/bscw.cgi/4231726

[MSW-NW 2012] Ministerium für Schule und Weiterbildung des Landes Nordrhein-Westfalen (Hrsg.): *Materialien zu den zentralen Abiturprüfungen im Fach Informatik ab 2012 Java-Version*. 2012. – http://www.standardsicherung.schulministerium.nrw.de/abitur-gost/getfile.php?file=3177 – geprüft: 01. März 2014

[MYSQL 2013] : *MySQL 5.1 Referenzhandbuch*. November 2013. – URL http://dev.mysql.com/doc/refman/5.1/de/index.html

Todo list