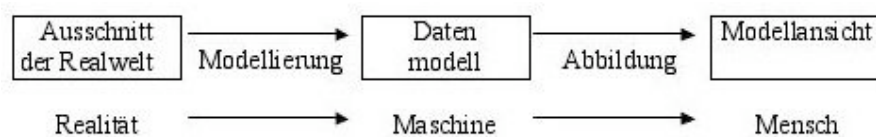


## 2 Grundlagen in MySQL und phpMyAdmin

**Bemerkung:**

Da die Schüler der 9. Jahrgangsstufe sich mit Datenbanken und Datenbankabfragen beschäftigt haben, werde ich in diesem Kapitel nur so weit die Grundlagen in MySQL und phpMyAdmin ansprechen, wie sie für Datenbankzugriffe mit Java in diesem Projekt benötigt werden. Leser, die bereits Erfahrung in der Modellierung von Datenbanken und in der Verwendung von phpMyAdmin und MySQL besitzen, können dieses Kapitel überspringen.

Für den Aufbau einer Datenbank muss zuerst ein Datenmodell definiert werden, also ein möglichst exaktes Abbild eines realen Weltausschnitts.

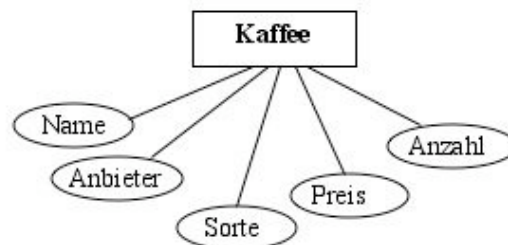


**Abbildung 2.1:** Von der Realwelt zur Datenbank

Dieses Datenmodell wird dann in einer grafischen Benutzeroberfläche (GUI) dem Anwender präsentiert. Der Programmierer stellt also in bestimmten Ansichten (Formularen) dem Anwender eine bestimmte Sicht der Daten dar.

### 2.1 Die Datenbank Kaffeehandel1

Du hast in deinen Regalen verschiedene Kaffeesorten stehen. Diese werden von ein einigen Firmen geliefert. Die Abbildung 2.2 zeigt eine erste, vereinfachte Informationsstruktur



**Abbildung 2.2:** Erste Informationsstruktur der Klasse *Kaffee*

Die zentrale Klasse der Datenbank *Kaffeehandel1* ist die Klasse *Kaffee*. Diese hat die Attribute Name, Anbieter, Sorte, Preis und Anzahl. Die Attribute der Objekte dieser Klasse *Kaffee* werden in der Datenbank in einer Tabelle gespeichert. Die Zeilen dieser Tabelle geben die Attributwerte eines bestimmten Objekts wider. Die Spalten dieser Tabelle entsprechen den Attributen und sollten daher die gleichen Namen haben.

### 2.1.1 Erstellen von Kaffeehandel1

Deine erste Datenbank *Kaffeehandel1* wird noch recht einfach gestaltet sein und besteht vorerst nur aus einer Tabelle *Kaffee*.

Die Abbildungen 2.3 und 2.4 zeigen dir, wie in *phpMyAdmin* die Datenbank *Kaffeehandel1* und die Tabelle *Kaffee* angelegt werden:



Abbildung 2.3: Anlegen der Datenbank *Kaffeehaus1*

Für das Erstellen der Klasse *Kaffee* benötigst du also eine Tabelle mit 6 Feldern. Das erste Feld beinhaltet den Primärschlüssel *KaffeeID*. Als Typ gibst du *INT* und als Extras *auto-increment* an. Somit musst du für *KaffeeID* keinen Wert eingeben, die Datenbank nimmt automatisch immer den nächsten Wert. Zusätzlich musst du den Radiobutton *Primärschlüssel* anwählen.



Abbildung 2.4: Anlegen der Tabelle *Kaffee*

Anschließend werden die restlichen Attribute als Feldnamen eingegeben. Einige Datentypen benötigen eine maximale Länge bzw. die Anzahl der Vor- und Nachkommastellen. Die entsprechenden Werte kannst du aus der Abbildung 2.4 entnehmen. Das Attribut Anzahl erhält als Standardwert 0.

Zur Verwaltung von Fremdschlüsseln und zur Überprüfung der referenziellen Integrität der Daten solltest du den Datenbanktyp *InnoDB* wählen.

Mit Speichern wird anschließend die Tabelle *Kaffee* in der Datenbank *Kaffeehandel1* angelegt.

Kaffee	
KaffeeID	
Name	
Sorte	
Preis	
anzahl	
Anbieter	

Feld	Typ
<u>KaffeeID</u>	Int(11)
Name	varchar(100)
Sorte	varchar(100)
Preis	decimal(3,2)
Anzahl	Int(11)
Anbieter	varchar(100)

Abbildung 2.5: Die Klasse *Kaffee*

Nun hast du die Vorbereitungen abgeschlossen und kannst anschließend Objekte der Klasse *Kaffee* erzeugen. Alle diese Objekte haben die in Abbildung 2.5 dargestellten Attribute, unterscheiden sich in ihren Attributwerten.

Feld	Typ	Funktion	Null	Wert
KaffeeID	Int(11)			
Name	varchar(100)			Espresso dOro
Sorte	varchar(100)			Espresso
Preis	decimal(3,2)			7.30
Anzahl	Int(11)			0
Anbieter	varchar(100)			Dallmayr

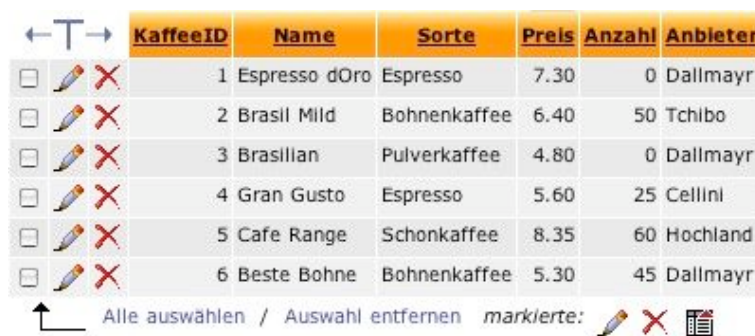
Abbildung 2.6: Erzeugung eines Objekts der Klasse *Kaffee*













Zur Erzeugung von Objekten der Klasse *Kaffee* wählst du die Registerkarte *Einfügen* und ordnest den Attributen die entsprechenden Attributwerte zu.

### Bemerkung:

In objektorientierten Programmiersprachen wie Java haben diese Objekte eindeutige Bezeichner. In einer Tabelle sind aber diese Bezeichner nicht mehr vorhanden. Daher können die verschiedenen Objekte ausschließlich nach den Werten ihrer Attribute unterschieden werden. Es muss also mindestens ein Attribut existieren, in dem sich alle Objekte unterscheiden. Dieses Attribut (oder auch eine Gruppe von Attributen) bezeichnet man als **Schlüssel**.

Diese Objekte werden in relationalen Datenbanken als Zeilen einer Tabelle dargestellt.



	KaffeeID	Name	Sorte	Preis	Anzahl	Anbieter
<input type="checkbox"/>  	1	Espresso dOro	Espresso	7.30	0	Dallmayr
<input type="checkbox"/>  	2	Brasil Mild	Bohnenkaffee	6.40	50	Tchibo
<input type="checkbox"/>  	3	Brasilian	Pulverkaffee	4.80	0	Dallmayr
<input type="checkbox"/>  	4	Gran Gusto	Espresso	5.60	25	Cellini
<input type="checkbox"/>  	5	Cafe Range	Schonkaffee	8.35	60	Hochland
<input type="checkbox"/>  	6	Beste Bohne	Bohnenkaffee	5.30	45	Dallmayr




Alle auswählen / Auswahl entfernen markierte:   

Abbildung 2.7: Die Zeilen stellen Objekte in der Tabelle *Kaffee* dar

Du hast nun in MySQL eine kleine Datenbank *Kaffeehandell* erstellt, die vorerst aus einer einzigen Tabelle *Kaffee* besteht. Beachten solltest du allerdings, dass diese Datenbank nicht den Normalformen genügt. Trotzdem kannst du ersten Abfragen an diese Datenbank richten.

## 2.1.2 Datenbankabfragen an Kaffeehandell

### 2.1.2.1 Projektion

Die Projektion ist eine einfache Datenbankabfrage, die nichts anderes macht, als bestimmte Attribute einer Tabelle auszuwählen. Im Prinzip stellt die Tabelle *Kaffee* die allgemeinste Form einer Projektion dar, da hier alle Attribute mit dem Operator „\*“ abgefragt wurden. Die Projektion ist oft erforderlich, da in der Regel nicht immer alle Eigenschaften eines Objekts ausgewertet bzw. aufgelistet werden müssen.

## Projektion

TeilnehmerID	Nachname	Vorname	Strasse	Wohnort
1	Sorglos	Susi	Hauptstr. 142	40210 Sonnstetten
2	Hofmann	Helma	Am Bächle 3	66822 Heidelberg
3	Hauer	hans	Im Winkl 16a	12329 Talhausen
4	Pfeiffer	Claudia	Mozartweg 6	74121 Ludwigshafen
5	Wallung	Walter	Panoramapfad 33	09663 Grünstadt
6	Peters	Paul	Am Markt 1	53522 Köln
7	Unterländer	Elke	Max-Weber-Str. 12	81023 München
8	Nicks	Steffi	Holzweg 8	73124 Oberndorf
9	Hirsch	Harry	Baumgartenstr. 2	75175 Pforzheim
*	(AutoWert)			

**Abbildung 2.8:** Projektion bedeutet Auswahl von *Attributen* einer Tabelle

Syntax:       **SELECT <Spalte>**  
                   **FROM <Tabelle>**

Um eine Abfrage an die Datenbank *Kaffeehandell* zu stellen, wählst du die Registerkarte SQL und gibst hier die entsprechende SQL-Anweisung ein.



**Abbildung 2.9:** Eingabe einer SQL-Anweisung

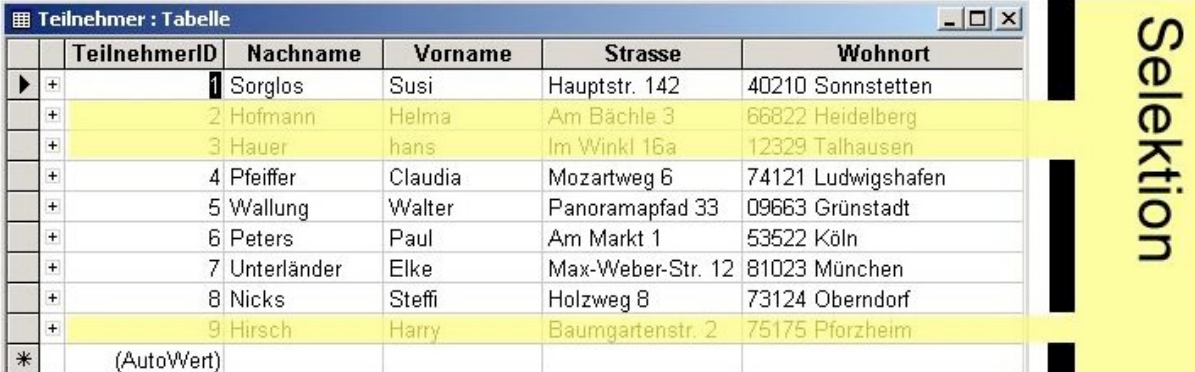
### Übung 2.1:

- Gesucht sind alle Daten außer KaffeeID von allen Kaffeesorten.
- Gesucht sind Name und Preis aller Kaffees sortiert nach Preis. (Verwende für die Sortierung `ORDER BY` in Verbindung mit `ASC` bzw. `DESC`.)
- Gesucht sind Name und Sorte aller Kaffees sortiert nach Sorte.

- d) Gesucht sind alle Anbieter.  
(Verwende **SELECT DISTINCT**, um Duplikate zu vermeiden.)
- e) Gesucht sind alle Sorten.

### 2.1.2.2 Selektion

Die Selektion stellt eine der wichtigsten Datenbankoperationen dar. Sie wird eingesetzt, um aus einer Tabelle die Menge der Datensätze herauszufiltern, die einer bestimmten Bedingung genügen.



	TeilnehmerID	Nachname	Vorname	Strasse	Wohnort
+	1	Sorglos	Susi	Hauptstr. 142	40210 Sonnstetten
+	2	Hofmann	Helma	Am Bächle 3	66822 Heidelberg
+	3	Hauer	hans	Im Winkl 16a	12329 Talhausen
+	4	Pfeiffer	Claudia	Mozartweg 6	74121 Ludwigshafen
+	5	Wallung	Walter	Panoramapfad 33	09663 Grünstadt
+	6	Peters	Paul	Am Markt 1	53522 Köln
+	7	Unterländer	Elke	Max-Weber-Str. 12	81023 München
+	8	Nicks	Steffi	Holzweg 8	73124 Oberndorf
+	9	Hirsch	Harry	Baumgartenstr. 2	75175 Pforzheim
*	(AutoWert)				

**Abbildung 2.10:** Selektion bedeutet Auswahl von *Datensätzen* einer Tabelle

Syntax:       **SELECT** <Spalte>  
                   **FROM** <Tabelle>  
                   **WHERE** <Bedingung>

#### Übung 2.2:

- a) Gesucht sind alle Daten des Kaffees „Gran Gusto“.
- b) Gesucht sind alle Kaffees des Anbieters „Dallmayr“.
- c) Gesucht sind alle Kaffees der Sorte „Bohnenkaffee“.
- d) Gesucht sind alle Kaffees, von denen weniger als 30 Packungen im Regal stehen.
- e) Gesucht sind alle Kaffees, die bereits ausverkauft sind.
- f) Gesucht sind alle Kaffees, die weniger als 6.00 Euro kosten.

## 2.1.3 Datenbankänderungen von Kaffeehandel1

### 2.1.3.1 Ändern von Datensätzen

Du willst nun als Besitzer des Kaffeehandels nach einer erfolgreichen Woche in deiner Datenbank *Kaffeehandel1* die Anzahl der Packungen, die noch in den Regalen stehen, eintragen. Das bedeutet, dass du nun bereits bestehende Datensätze ändern muss.

```
Syntax:      UPDATE <Tabelle>
              SET Spalte1 = Wert1, Spalte2 = Wert2, ...
              WHERE <Bedingung>
```

Zum Beispiel ändert die SQL-Anweisung an der Datenbank *Kaffeehandel1*

```
UPDATE Kaffee
SET Anzahl = 75
WHERE Name LIKE 'Espresso dOro'
```

in der Tabelle *Kaffee* den entsprechenden Datensatz ein.

#### Übung 2.3:

- Ändere die Anzahl der Packungen von „Espresso dOro“ auf 75.
- Der Kaffee „Gran Gusto“ findet reißenden Absatz fand. Du erhöhst deshalb den Preis auf 6.10 Euro.
- Der Kaffee „Cafe Range“ wird zum Ladenhüter. Du erniedrigst deshalb den Preis auf 7.90 Euro.
- Du hast festgestellt, dass der Kaffee „Brasil Mild“ auch ein Schonkaffee ist. Ändere deine Datenbank.

### 2.1.3.2 Einfügen von Datensätzen

Im Sortiment hast du nun den neuen Kaffee aufgenommen mit folgenden Daten:

Namen	Colombian
Anbieter	Tchibo
Sorte	Pulverkaffee
Preis	5.25

Syntax: **INSERT INTO <Tabelle> [<Spalte1>,<Spalte2>,...]  
VALUES (<Wert1>, <Wert2>, ...)**

Die Eingabe der SQL-Anweisung

```
INSERT INTO Kaffee  
VALUES ('Colombian', 'Pulverkaffee', 5.25, 'Tchibo')
```

führt jedoch zu folgender Fehlermeldung:



Abbildung 2.11: Fehlermeldung bei der INSERT-Anweisung

denn die Tabelle *Kaffee* besteht aus sechs Datenfeldern mit zum Teil voreingestellten Werten:

KaffeeID		auto_increment
Name	Colombian	
Sorte	Pulverkaffee	
Preis	7.25	
Anzahl		0
Anbieter	Tchibo	

Nur wenn du genauso viele Werte übergibst, wie die Tabelle Datenfelder besitzt, darfst du die Attributliste in der SQL-Anweisung weglassen.

In dieser Tabelle *Kaffee* gilt, dass du *KaffeeID* mit einer *auto-increment*-Option versehen hast und für *Anzahl* einen DEFAULT-Wert angegeben hast. Deshalb musst du alle Datenfelder außer *KaffeeID* und *Anzahl* in der Attributliste angeben.

```
INSERT INTO Kaffee(Name, Sorte, Preis, Anbieter)
VALUES ('Colombian', 'Pulverkaffee', 7.25, 'Tchibo')
```

In diesem Fall vergibt die Datenbank automatisch die korrekte *KaffeeID* und setzt den Wert von *Anzahl* auf 0.



	KaffeeID	Name	Sorte	Preis	Anzahl	Anbieter
<input type="checkbox"/>	1	Espresso dOro	Espresso	7.30	0	Dallmayr
<input type="checkbox"/>	2	Brasil Mild	Bohnenkaffee	6.40	50	Tchibo
<input type="checkbox"/>	3	Brasilian	Pulverkaffee	4.80	0	Dallmayr
<input type="checkbox"/>	4	Gran Gusto	Espresso	5.60	25	Cellini
<input type="checkbox"/>	5	Cafe Range	Schonkaffee	8.35	60	Hochland
<input type="checkbox"/>	6	Beste Bohne	Bohnenkaffee	5.30	45	Dallmayr
<input type="checkbox"/>	7	Colombian	Pulverkaffee	7.25	0	Tchibo

Abbildung 2.12: Der neue Kaffee wurde eingefügt

## 2.2 Die Datenbank Kaffeehandel2

In deiner Datenbank willst du nun weitere Details über die Anbieter speichern. Mögliche Datenfelder der Klasse Anbieter zeigt Abbildung 2.13.

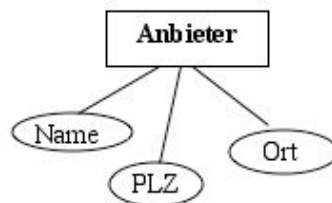
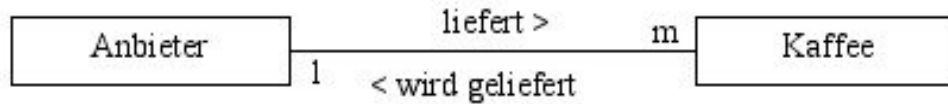


Abbildung 2.13: Erste Informationsstruktur der Klasse *Anbieter*

Zusätzlich zur Klasse *Kaffee* benötigst du nun die Klasse *Anbieter*. Diese hat die Attribute *Name*, *PLZ* und *Ort*. Die Attribute der Objekte dieser Klasse *Anbieter* werden in der Datenbank in einer Tabelle gespeichert. Die Zeilen dieser Tabelle geben die Attributwerte eines bestimmten Objekts wider. Die Spalten dieser Tabelle entsprechen den Attributen und sollten daher die gleichen Namen haben.

## 2.2.1 Erstellen von Kaffeehandel2

Zwischen den Klassen *Anbieter* und *Kaffee* besteht die in Abbildung 2.14 dargestellte Beziehung:



**Abbildung 2.14:** Klassendiagramm von *Kaffeehandel2*

Diese Beziehung wird wie folgt gelesen:

Eine bestimmter Anbieter liefert **m** (eine oder mehrere) Kaffees.

Eine bestimmter Kaffee wird geliefert von **1** (genau einem) Anbieter.

Es kann also jedes Objekt der Klasse *Anbieter* mit beliebig vielen Objekten der Klasse *Kaffee* in Beziehung treten. Jedoch kann jedes Objekt der Klasse *Kaffee* nur mit genau einem Objekt der Klasse *Anbieter* in Beziehung treten. Somit ist die Kardinalität dieser Beziehung 1:m.

Um die Klassenbeziehung *liefert/wird geliefert* zu realisieren, musst du den Primärschlüssel der Klasse *Anbieter* als Fremdschlüssel in der Klasse *Kaffee* angeben. Der Schlüssel des Objekts auf der Seite mit „1“ (*Anbieter.AnbieterID*) wird als Attribut beim Objekt auf der Seite mit „m“ (*Kaffee.AnbieterNr*) gespeichert.

Somit enthält die Datenbank *Kaffeehandel2* nun die folgenden beiden Klassen:

Feld	Typ
<u>AnbieterID</u>	int(11)
Name	varchar(100)
PLZ	varchar(5)
Ort	varchar(100)

**Abbildung 2.15:** Die Klasse *Anbieter*

Kaffee
KaffeeID
Name
Sorte
Preis
Anzahl
AnbieterNr

Feld	Typ
<u>KaffeeID</u>	Int(11)
Name	varchar(100)
Sorte	varchar(100)
Preis	decimal(3,2)
Anzahl	Int(11)
AnbieterNr	Int(11)

Abbildung 2.16: Die Klasse *Kaffee*

Im letzten Schritt musst du noch die referenzielle Integrität der Daten gewährleisten, d.h. für jeden Fremdschlüsselwert muss auch ein Datensatz mit diesem Schlüsselwert als Primärschlüssel existieren.

Bei MySQL können Fremdschlüssel nur vereinbart werden, wenn für beide Tabellen als Datenbankverwaltungsprogramm *InnoDB* angegeben ist. Aus Geschwindigkeitsgründen muss der Fremdschlüssel eines Index haben.

Feld	Typ	Kollation	Attribute	Null	Standard	Extra	Aktion
<input type="checkbox"/> <u>KaffeeID</u>	Int(11)			Nein		auto_increment	[edit] [delete] [index] [primary] [foreign] [unique]
<input type="checkbox"/> Name	varchar(100)	utf8_general_ci		Nein			[edit] [delete] [index] [primary] [foreign] [unique]
<input type="checkbox"/> Sorte	varchar(100)	utf8_general_ci		Nein			[edit] [delete] [index] [primary] [foreign] [unique]
<input type="checkbox"/> Preis	decimal(3,2)			Nein			[edit] [delete] [index] [primary] [foreign] [unique]
<input type="checkbox"/> Anzahl	Int(11)			Nein 0			[edit] [delete] [index] [primary] [foreign] [unique]
<input type="checkbox"/> AnbieterNr	Int(11)			Nein			[edit] [delete] [index] [primary] [foreign] [unique]

Name	Typ	Kardinalität	Aktion	Feld
PRIMARY	PRIMARY	2	[edit] [delete]	KaffeeID
AnbieterNr INDEX	INDEX	2	[edit] [delete]	AnbieterNr

Abbildung 2.17: Das Datenfeld *AnbieterNr* wird indiziert

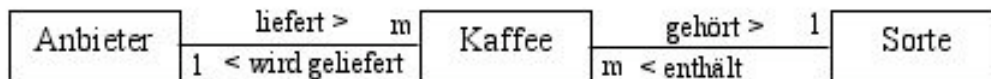
Die Vereinbarung von Fremdschlüsseln ist mit phpMyAdmin sehr einfach. Es wird die Tabelle ausgewählt, in der der Fremdschlüssel vereinbart werden soll.

Der Link *Beziehungsübersicht* führt zum Fenster, das in Abbildung 2.18 gezeigt wird.

**Abbildung 2.18:** Der Link *Beziehungsübersicht* führt zum Fenster *Verweise*

Auf dieser Seite kannst du über Popup-Menüs für alle Felder mit Index auswählen, ob und wenn ja welches Feld sie referenzieren sollen und welche Aktionen ausgeführt werden sollen, wenn der referenzierte Schlüssel gelöscht bzw. verändert wird.

Analog wie die Anbieterfirmen lassen sich auch die Kaffeesorten in einer eigenen Klasse verwalten. Das Klassendiagramm aus Abbildung 2.14 kann somit erweitert werden.



**Abbildung 2.19:** Klassendiagramm von *Kaffeehandel2*

Somit enthält die Datenbank *Kaffeehandel2* zusätzlich die weitere Klasse *Sorte*.

Sorte	<b>Feld</b>	<b>Typ</b>
SorteID	<u>SorteID</u>	Int(11)
Sorte	Sorte	varchar(100)

**Abbildung 2.20:** Die Klasse *Sorte*

## Übung 2.4:

Erweitere die Datenbank *Kaffeehandel2* um die Klasse *Sorte*. Achte die korrekte 1:m-Beziehung und die referenzielle Integrität zwischen den Klassen *Sorte* und *Kaffee*.

Nun hast du die Vorbereitungen abgeschlossen und kannst anschließend Objekte der Klassen *Anbieter*, *Kaffee* und *Sorte* erzeugen und diese in den entsprechenden Tabellen speichern.

### Übung 2.5:

Erzeuge nun die Objekte der jeweiligen Klassen und füge diese in die entsprechenden Tabellen ein. In Abbildung 2.21 bietet einen möglichen Vorschlag an.

AnbieterID	Name	PLZ	Ort
1	Dallmayr	80331	München
2	Cellini	22587	Hamburg
3	Tchibo	22297	Hamburg
4	Hochland	70597	Stuttgart

SorteID	Sorte
1	Espresso
2	Bohnenkaffee
3	Pulverkaffee
4	Schonkaffee

KaffeeID	Name	SorteNr	Preis	Anzahl	AnbieterNr
1	Espresso dOro	1	7.30	0	1
2	Brasil Mild	2	6.40	50	3
3	Brasillian	3	4.80	0	1
4	Gran Gusto	1	5.60	25	2
5	Cafe Range	4	8.35	60	4
6	Beste Bohne	2	5.30	45	1
7	Colombian	3	7.25	0	3

Abbildung 2.21: Objekte in den Tabellen *Anbieter*, *Sorte* und *Kaffee*

## 2.2.2 Datenbankabfragen an Kaffeehandel2

Ein Kunde fragt dich nun, in welcher Stadt die Anbieter aller von dir geführten Kaffeesorten ihren Stammsitz haben.

Abbildung 2.21 zeigt, dass zur Beantwortung dieser Frage die beiden Tabellen *Anbieter* und *Kaffee* benötigt werden. Aus den Datenfeldern *Anbieter.Ort* und *Kaffee.Name* kannst du den Stammsitz des Anbieters und den Namen des Kaffees ermitteln. Bei dieser Abfrage werden also die beiden Tabellen *Anbieter* und *Kaffee* benötigt. Diese werden in einem ersten Schritt zu einem Verbund, dem so genannten Kreuzprodukt, zusammengeführt.

### Übung 2.6:

a) Richte folgende SELECT-Anweisung an deine Datenbank *Kaffeehandel2*:

```
SELECT Kaffee.Name, Anbieter.Ort  
FROM Kaffee, Ort
```

Überlege, wie die recht mächtige Ergebnistabelle entstanden sein könnte.

b) Richte folgende SELECT-Anweisung an deine Datenbank *Kaffeehandel2*:

```
SELECT Kaffee.Name, Kaffee.AnbieterNr, Anbieter.AnbieterID,  
Anbieter.Ort  
FROM Kaffee, Anbieter
```

Überlege an Hand der Ergebnistabelle, nach welchen Datensätzen gefragt ist.

Die Beziehung zwischen den beiden Tabellen besteht über den Primärschlüssel *Anbieter.AnbieterID* und dem zugehörigen Fremdschlüssel *Kaffee.AnbieterNr*. Im zweiten Schritt musst du also nun diejenigen Objekte aus den Klassen *Anbieter* und *Kaffee* auswählen, bei denen Primärschlüssel und Fremdschlüssel den gleichen Wert besitzen. Das Ergebnis nach diesem zweiten Schritt bezeichnet man als (natürlichen) Verbund (*natural join*)

Abbildung 2.22 zeigt die SQL-Abfrage und das zugehörige Ergebnis.



Name	Ort
Espresso dOro	München
Brasilian	München
Beste Bohne	München
Gran Gusto	Hamburg
Brasil Mild	Hamburg
Colombian	Hamburg
Cafe Range	Stuttgart

Abbildung 2.22: Zeigt den Anbieter von jedem Kaffee

Übung 2.7:

- Welche Kaffees werden aus Hamburg geliefert?
- Zu jedem Kaffee soll der Name und der Ort des Anbieters angezeigt werden.
- Welche Kaffees werden von der Firma Dallmayr geliefert?
- Liste alle Pulverkaffees auf.
- usw.

## 2.3 Die Datenbank Kaffeehandel3

In deiner Datenbank möchtest du nun auch die Geschmacksrichtungen der einzelnen Kaffees verwalten. Will ein Kunde von dir wissen, welcher Kaffee den Geschmack „mild“ haben, so sollten aus deiner Datenbank alle Kaffees mit dieser entsprechenden Geschmacksrichtung aufgelistet werden.

Du wirst nun eine neue, recht einfach gehaltene Datenbank *Kaffeehandel3* erstellen, in der die Beziehung zwischen den einzelnen Kaffees und deren Geschmacksrichtung dargestellt wird

### 2.3.1 Erstellen von Kaffeehandel3

Hierfür benötigst du zunächst die beiden Klassen *Kaffee* und *Geschmack*

Die Klasse *Kaffee* besitzt die Attribute *KaffeeID* und *Name*.



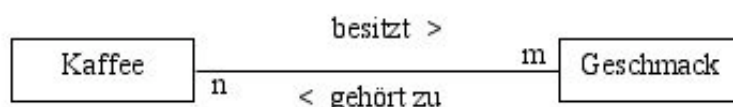
Abbildung 2.23: Die Klasse *Kaffee*

Analog ist die Klasse *Geschmack* aufgebaut. Sie hat die Attribute *GeschmackID* und *Art*



Abbildung 2.24: Die Klasse *Geschmacke*

Zwischen den Klassen *Kaffee* und *Geschmack* besteht die in Abbildung 2.25 dargestellte Beziehung.



### Abbildung 2.25: Klassendiagramm von *Kaffeehandl3*

Diese Beziehung wird wie folgt gelesen:

Eine bestimmte Kaffeesorte hat **m** (eine oder mehrere) Geschmacksrichtungen.  
Eine bestimmte Geschmacksrichtung gehört zu **n** (einer oder mehreren) Kaffeesorten.

Es kann also jedes Objekt der Klasse *Kaffee* mit beliebig vielen Objekten der Klasse *Geschmack* in Beziehung treten. Ebenso kann aber auch jedes Objekt der Klasse *Geschmack* mit beliebig vielen Objekten der Klasse *Kaffee* in Beziehung treten. Somit ist die Kardinalität dieser Beziehung n:m.

Um die Klassenbeziehung *hat einen/gehört zu* zu realisieren, genügt es nun nicht mehr wie bei einer 1:m-Beziehung, den Primärschlüssel einer Klasse als Fremdschlüssel in einer anderen Klasse anzugeben. Du musst die Beziehung zwischen den Klassen *Kaffee* und *Geschmack* als separate Zuordnungstabelle *Kaff\_Geschm* erfassen.

Kaff_Gechm	
KaffeeNr	
GeschmackNr	

Feld	Typ
<u>KaffeeNr</u>	Int(11)
<u>GeschmackNr</u>	Int(11)

Abbildung 2.26: Die Klasse *Kaff\_Geschm*

Diese Tabelle *Kaff\_Geschm* enthält also Objekte, die jedem Fremdschlüssel *KaffeeNr* einen bestimmten Fremdschlüssel *GeschmackNr* zuordnet.

Somit kannst du durch Verknüpfung der drei Tabellen *Kaffee*, *Kaff\_Geschm* und *Geschmack* genau zuordnen, welcher Kaffee welche Geschmacksrichtungen besitzt.

Allerdings hat sie eine bisher noch nie genutzte Eigenschaft. Der Schlüssel für die Tabelle *Kaff\_Geschm* kann weder *KaffeeNr* noch *GeschmackNr* sein, da beide Attribute mehrmals den gleichen Wert besitzen. Nur das Paar (*KaffeeNr*, *GeschmackNr*) ist eindeutig. Hier verwendest du zum ersten Mal einen aus zwei Attributen zusammengesetzten Schlüssel.

Im letzten Schritt musst du noch die referenzielle Inetgrität der Daten gewährleisten, d.h. dass für jeden Fremdschlüsselwert auch ein Datensatz mit diesem Schlüsselwert existiert.

Bei MySql können Fremdschlüssel nur vereinbart werden, wenn für beide Tabellen als Datenbankverwaltungsprogramm *InnoDB* angegeben ist. Aus

Geschwindigkeitsgründen muss der Fremdschlüssel eines Index haben. Da in diesem Fall die Kombination der beiden Fremdschlüssel den Primärschlüssel bildet, sind diese beiden bereits automatisch indiziert.

Die Vereinbarung von Fremdschlüsseln ist mit phpMyAdmin sehr einfach. Es wird die Tabelle ausgewählt, in der der Fremdschlüssel vereinbart werden soll. Der Link *Beziehungsübersicht* führt zum Fenster, das in Abbildung 5.8 gezeigt wird.

Abbildung 2.27: Der Link *Beziehungsübersicht* führt zu den *Verweise*

Auf dieser Seite kannst du über Popup-Menüs für alle Felder mit Index auswählen, ob und wenn ja welches Feld sie referenzieren sollen und welche Aktionen ausgeführt werden sollen, wenn der referenzierte Schlüssel gelöscht bzw. verändert wird.

Die Attribute der Objekte der Klasse *Kaffee*, *Geschmack* und *Kaff-Geschm* werden in der Datenbank in den Zeilen der entsprechenden Tabelle gespeichert.

KaffeeID	Name
1	Classico
2	Colomblan
3	Mocca Gold

GeschmackID	Art
1	cremig
2	vollmundlg
3	mild

KaffeeNr	GeschmackNr
1	2
1	3
2	1
2	2
3	2
3	3

Abbildung 2.28: Objekte in den Tabellen *Kaffee*, *Geschmack* und *Kaff\_Geschm*

## Übung 2.8:

Lege eine neue Datenbank *Kaffeehandel3* an. Erzeuge in dieser die Klassen *Kaffee*, *Geschmack* und *Kaff\_Geschm*. Achte auf die referenzielle Integrität. Nun füge die in den Abbildung 2.28 dargestellten Datensätze ein.

Abbildung 2.29 zeigt die eingangs gestellte Frage nach allen Kaffees, die die Geschmacksrichtung „mild“ besitzen. Solche Abfragen bei n:m-Beziehungen müssen alle drei Tabellen verwenden.

Zeige Datensätze 0 - 1 (2 Insgesamt, die Abfrage dauerte 0.0019 sek)

**SQL-Befehl:**  

```
SELECT Kaffee.Name, Geschmack.Art
FROM Kaffee, Kaff_Geschm, Geschmack
WHERE (
  Kaffee.KaffeeID = Kaff_Geschm.KaffeeNr
)
AND (
  Kaff_Geschm.GeschmackNr = geschmack.GeschmackID
)
AND (
  Geschmack.Art = 'mild'
)
LIMIT 0 , 30
```

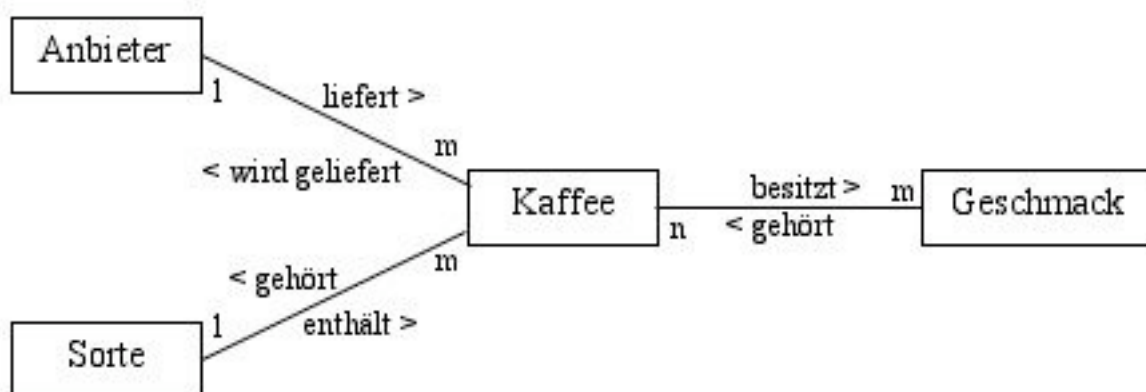
[ Bearbeiten ]
[ SQL erklären ]
[ PHP-Code erzeugen ]
[ Aktualisieren ]

Name	Art
Classico	mild
Mocca Gold	mild

**Abbildung 2.29:** Zeigt alle Kaffees mit der Geschmacksrichtung „mild“

## 2.4 Die Datenbank *Kaffeehandel4*

Die Datenbank *Kaffeehandel4* vereinigt nun alle drei bisher besprochenen Datenbanken.



**Abbildung 2.30:** Klassendiagramm von *Kaffeehandel4*

Dem interessierten Leser bleibt es nun überlassen, diese Datenbank zu erstellen und Datensätze einzufügen. Weitere Informationen und Übungen findest du in

den entsprechenden Büchern bzw. in meinem Skript *Datenmodellierung und Datenbanksysteme, Eine praxisnahe Einführung mit ACCESS*

## 2.5 Kopieren einern Datenbank

In den folgenden Kapiteln wirst du von Java aus Datenbankzugriffe auf die Datenbanken kennen lernen. Hier erweist es sich als sinnvoll, beim Üben nicht mit den Originalen *Kaffeeladenx* zu arbeiten. Deswegen wirst du nun lernen, von der Datenbank *Kaffeeladen1* eine Kopie *Kaffeehaus1* anzulegen.

Zum Erzeugen einer Kopie rufst du die Registerkarte *Operationen* auf.

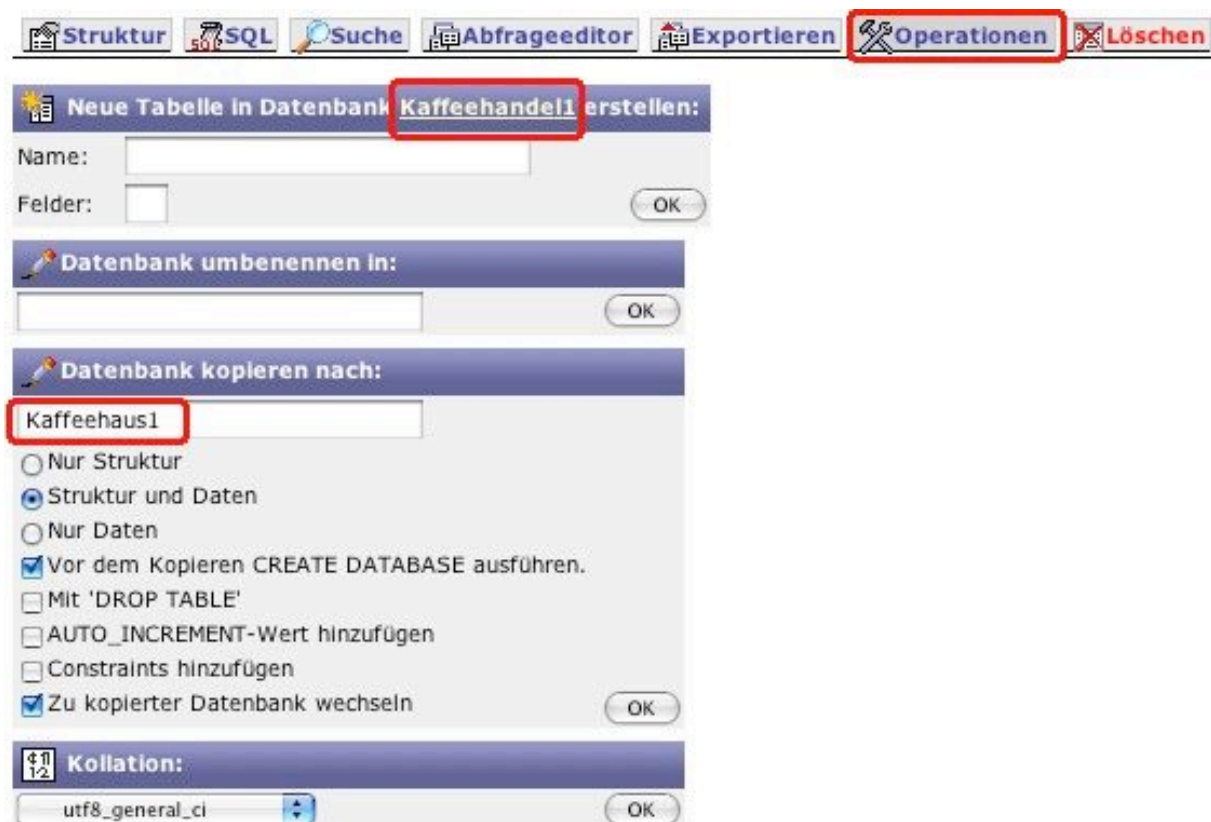


Abbildung 2.31: Kopieren von *Kaffeeladen1* nach *Kaffeehaus1*

Achte darauf, dass du von *Kaffeehandell* eine Kopie erstellst. Dieser Kopie gibst in das entsprechende Textfeld den Namen *Kaffeehaus1*. Kopiert werden sollen „Struktur und Daten“, somit erhältst du eine Eins-zu-Eins-Kopie. Anschließend wählst du die Optionen „Vor dem Kopieren CREATE DATABASE ausführen“ und, falls du mit der Zieldatenbank gleich weiterarbeiten willst die Option „Zu kopierter Datenbank wechseln“. Nach dem Drücken auf *OK* wird diese neue Datenbank aufgelistet.

## 2.6 Exportieren einer Datenbank

Die Exportfunktion wird in der Registerkarte Exportieren zur Verfügung gestellt.

The screenshot shows the 'Exportieren' (Export) dialog in phpMyAdmin. The 'Exportieren' tab is selected, and the 'SQL-Optionen' (SQL Options) section is expanded. The option 'Tabellen- und Feldnamen in einfachen Anführungszeichen' (Table and field names in simple quotes) is checked and highlighted with a red box. The 'Senden' (Send) checkbox is also checked and highlighted with a red box. The 'Dateinamenskonvention' (Filename convention) field is set to 'Kaffeehandel1' and is also highlighted with a red box. The 'Kompression' (Compression) section shows 'keine' (none) selected. The 'OK' button is visible at the bottom right.

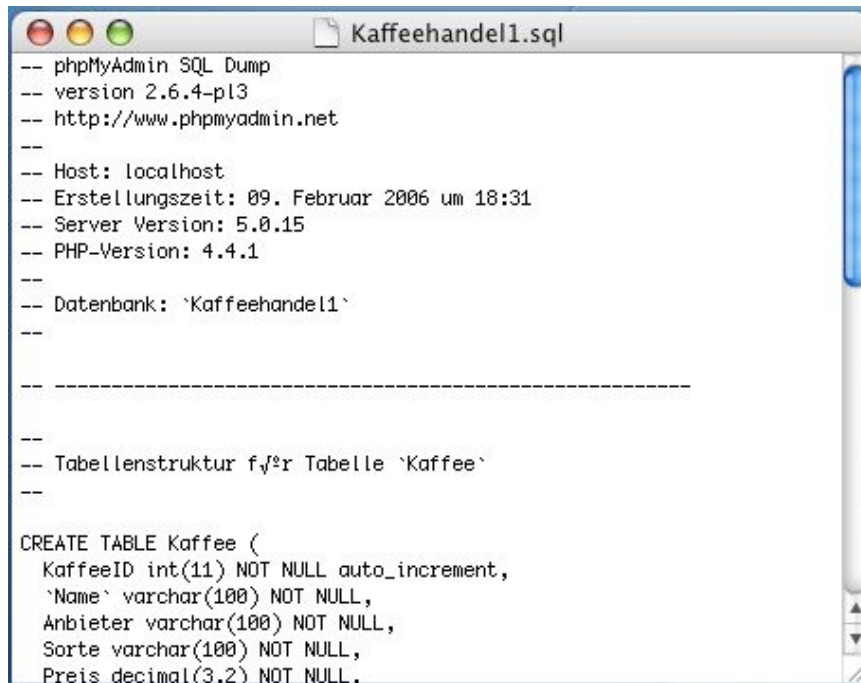
Zum Exportieren der Datenbank *Kaffeehandel1* können die Standardvorgaben beibehalten werden.

### Bemerkung:

Bei mir unter Mac OS X10.4 bereitete XAMPP (Version 0.4) beim Export auf ein Windows-Betriebssystem Probleme, wenn die Tabellen- und Feldnamen in einfache Anführungszeichen gesetzt wurden.

Soll diese Datenbank auf einem anderen System gespeichert werden, so musst du dies explizit im Exportformular angeben. Hierzu aktivierst du die Option *Senden*. Im zugehörigen Textfeld *Dateinamenskonvention* legst fest, wie die

Zieldatei heißen soll. Nach dem drücken auf *OK* wird im obigen Beispiel die Datenbank als *Kaffeeladen1.sql* gespeichert. Diese Datei kann dann mit Hilfe eines Texteditors gelesen werden.



```
-- phpMyAdmin SQL Dump
-- version 2.6.4-pl3
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Erstellungszeit: 09. Februar 2006 um 18:31
-- Server Version: 5.0.15
-- PHP-Version: 4.4.1
--
-- Datenbank: `Kaffeehandell1`
--
-----
--
-- Tabellenstruktur für Tabelle `Kaffee`
--
CREATE TABLE Kaffee (
  KaffeeID int(11) NOT NULL auto_increment,
  `Name` varchar(100) NOT NULL,
  Anbieter varchar(100) NOT NULL,
  Sorte varchar(100) NOT NULL,
  Preis decimal(3,2) NOT NULL,
```

Abbildung 2.32: Die Datei *Kaffeehandell1* im Texteditor

## 2.7 Importieren einer Datenbank

Die Importfunktion einer Datenbank liegt in phpMyAdmin etwas versteckt.



Abbildung 2.33: Button *SQL* und Button *Dateiimport*

In der linken Seitenleiste befindet sich der Button *SQL*. Klickst du auf diesen, so öffnet sich das Fenster zur Eingabe von SQL-Befehlen. Hier findest du auch den Button *Dateiimport*.



**Abbildung 2.34:** Auswahl der zu importierenden Datei

Nach dem Drücken auf diesen Button öffnet sich ein Fenster, in dem du nun die zu importierende Datei *neueDatenbank.sql* auswählen kannst. Anschließend wird diese neue Datenbank *neueDatenbank* in der linken Leiste unter den Datenbanken aufgelistet.