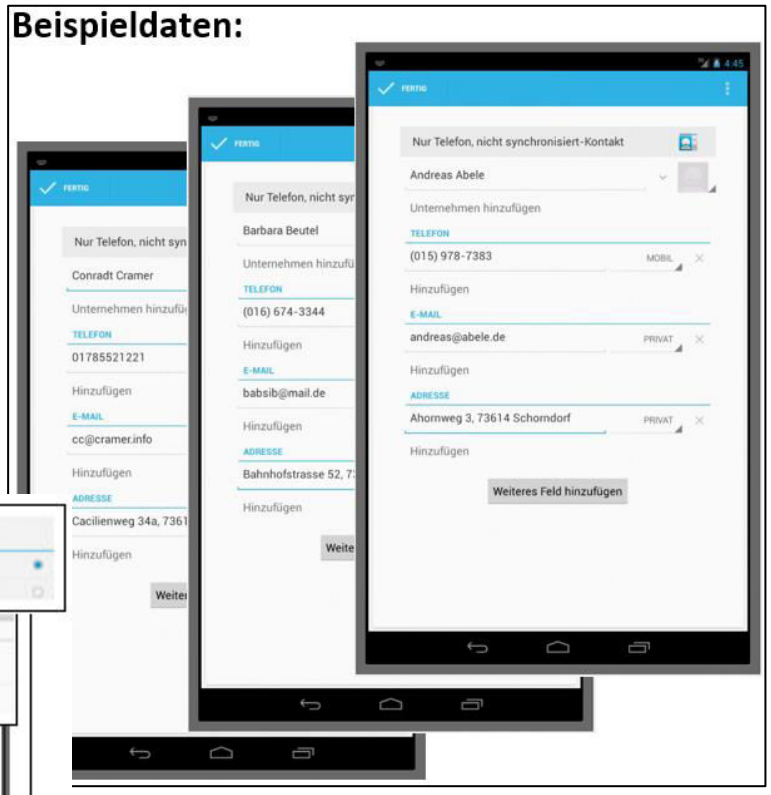
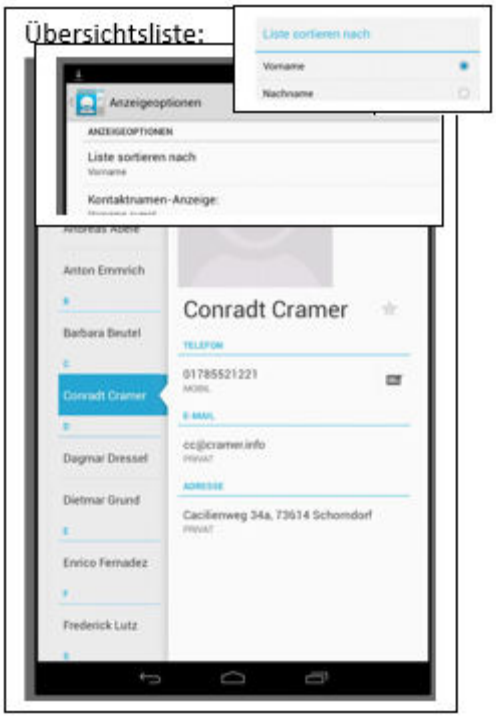
L1_1 Entwurf einer Tabelle erstellen – Aufgaben

Heiner Blechle hat sich seinen beruflichen Traum erfüllt: Er hat seine Fahrlehrerprüfung bestanden und macht sich nun mit einem gebrauchten Fahrschulauto selbstständig. Über seinen Sportverein hat er zu Jugendlichen und jungen Erwachsenen bereits Kontakte geknüpft, die bei ihm die Fahrschule besuchen möchten. Deren Namen und Kontaktdaten hat er vorläufig auf seinem Smartphone gespeichert:





Heiner Blechle stellt bald fest, dass die Speicherung der Daten seiner Kunden allein auf dem Handy unpraktisch ist. So braucht er diese Informationen regelmäßig für Meldungen an die Führerscheinstelle bei der Kreisverwaltung, den TÜV für Fahrprüfungen aber auch für die Erstellung von Rechnungen. Außerdem plant er, zu Werbezwecken entsprechende Infopost zu verschicken.

Alexander, ein Freund von Heiner aus dem Sportverein, studiert Wirtschaftsinformatik.
„Wenn du die Daten deiner Kunden nicht nur zum Telefonieren nutzen willst, brauchst du eine Datenbank. Das ist auch gar nicht so schwierig. Du musst nur eine Datenbanktabelle erstellen und die Daten, die du im Smartphone gespeichert hast, dort erfassen. Für den Entwurf einer Datenbanktabelle orientierst du dich am besten an folgendem Entwurfsraster."

Entwurfsraster einer Tabelle

Tabellenname:								
Attributname								
Datentyp								
max. Zeichenanzahl								

Aufgaben:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informationsmaterial *L1_1 Information Tabellenentwurf.docx.*

1 Formulieren Sie jeweils eine Definition für folgende Begriffe:

- Datensatz
- Attribut
- Primärschlüssel
- Attributwert

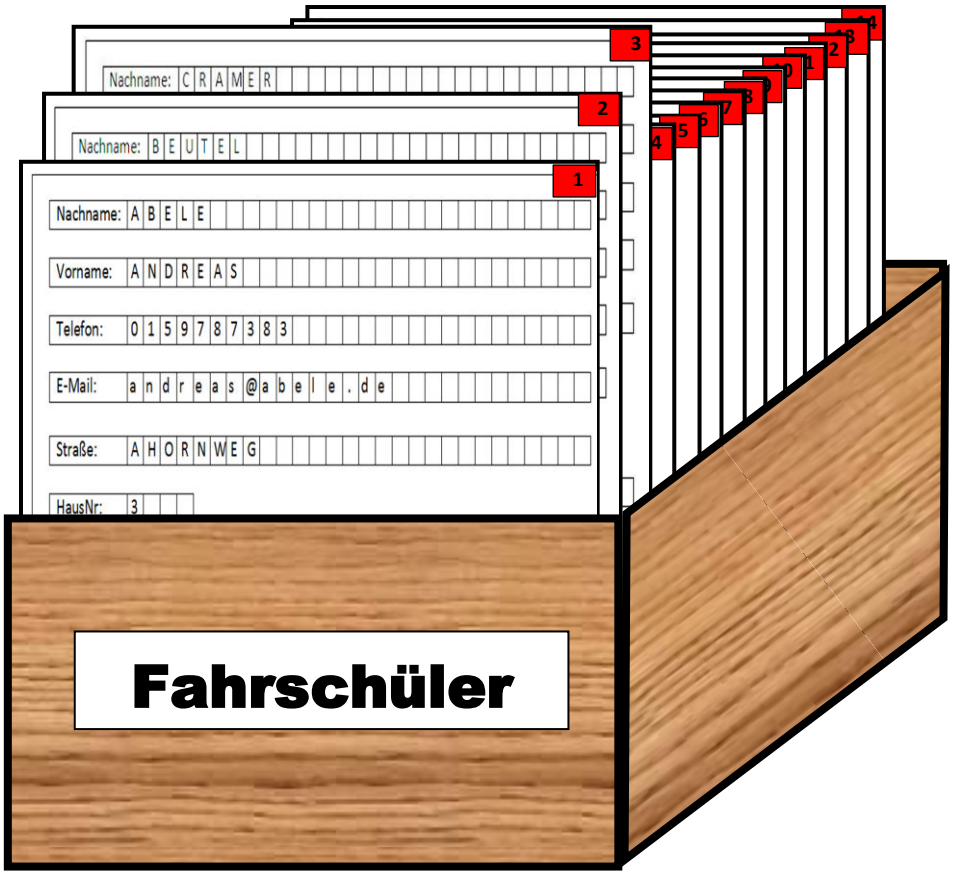
2 Erstellen Sie für Heiner Blechle nach dem abgebildeten Entwurfsraster einen Tabellenentwurf für die Datenbanktabelle *fahrschueler*.
Verwenden Sie die Entwurfsvorlage *L1_1 Vorlage Tabellenentwurf.docx*.

L1_1 Struktur einer Datenbanktabelle entwerfen – Information

Alexander, ein Freund aus dem Sportverein, studiert Wirtschaftsinformatik. Dieser erklärt ihm, dass man früher – im Vor-EDV-Zeitalter – die Daten auf vorgefertigte Karteikarten in das dafür vorgesehene Raster geschrieben und diese dann in einen Karteikasten einsortiert und aufbewahrt hat.

1

Nachname:	**ABELE**
Vorname:	**ANDREAS**
Telefon:	**01597873833**
EMail:	**andreas@abele.de**
Straße:	**AHORNWEG**
HausNr:	**3**
PLZ:	**73614**
Ort:	**SCHORNDORF**



In Datenbanken werden die Daten in Tabellen erfasst. Eine Datenbanktabelle enthält Informationen, die in einem sachlogischen Zusammenhang stehen.

Z.B.: Kundendaten → Tabelle *kunden*; – Daten der Vereinsmitglieder → Tabelle *mitglieder*
(Für die Schreibweise der Tabellennamen gilt die Konvention:

➔ Kleinbuchstaben, Plural, keine Umlaute/Sonderzeichen, keine Leerstellen

Zu beachten ist, dass Tabellen aus Zeilen und Spalten bestehen. Für sie gelten folgende Regeln:

- Jede Spalte enthält eine konkrete Eigenschaft (Fachbegriff Attribut)
  Z.B.: Nachname eines Kunden, Geburtsdatum eines Vereinsmitglieds, etc.
- Jeder Eintrag zu einer Eigenschaft wird als Attributwert bezeichnet. Z.B.: Huber, 2002-05-27
- Jedes Attribut muss einwertig (atomar) sein, d.h. jedes Attribut enthält nur einen Wert.
  Z.B.: Straße und Hausnummer, Vorname und Nachname
- Jede Spalte hat einen eindeutigen Namen (Fachbegriff Attributname)
  Z.B.: *nachname*, *geburtsdatum*, etc.
- Für jedes Attribut muss festgelegt werden, ob darin Texte (Buchstaben, Ziffern oder Sonderzeichen) oder Zahlenwerte (wenn damit gerechnet werden soll), gespeichert werden, d.h. es muss der jeweilige Datentyp festgelegt werden.
  (siehe Erläuterungen unten)
- Jede Zeile repräsentiert einen Eintrag. Jeder Eintrag hat die gleichen Attribute. Einträge werden auch als **Datensätze** (Tupel) bezeichnet.
  Z.B.: Alle Informationen zu einem Kunden.
- Damit jeder Eintrag eindeutig identifiziert werden kann, ist in jeder Tabelle ein sogenannter Primärschlüssel festzulegen. Dies kann der Eintrag einer Spalte sein, wenn dieser eindeutig ist (z.B. *personalausweisnr*). I.d.R. wird jedoch eine spezielle Schlüsselnummer als künstlicher Schlüssel eingefügt (z.B. *kundennr*). Sie sollte den Datentyp INT erhalten.
- Eine Tabelle wird in der Fachsprache auch als Relation bezeichnet.

Datentypen (Auszug):

Typ	Bedeutung	Speicherplatz	Beispiel	Hinweis
INT	Integer = Ganze Zahlen von -2.147.483.648 bis +2.147.483.647	4 Bytes	Blutspenden: 15	
DOUBLE	Kommazahlen von $-1{,}798\times10^{308}$ bis $+1{,}798\times10^{308}$	8 Bytes	Preis: 17.99	Dezimalpunkt (nicht Komma)
VARCHAR(n)	Zeichenkette mit variabler Länge bis zu n Zeichen	n + 1 Byte	Nachname: Häberle	n legt die max. Zeichenanzahl des Textes fest.
DATE	Datum von 01.01.1000 bis 31.12.9999	3 Bytes	Geburts-Datum: 1991-11-30	YYYY-MM-DD
TIME	Zeit	3 Bytes	Zielankunft: 03:56:04	hh:mm:ss
BOOLEAN TINYINT(1)	Wahrheitswert	1 Bytes	Anmeldung: true / false Bestanden: 1 / 0	Die MySQL Workbench verwendet als Wahrheitswert den Typ TINYINT(1)

Tabellenentwurf

Tabellenname:									
Attributname									
Datentyp									
max. Zeichen-anzahl									
Primär-schlüssel	☐	☐	☐	☐	☐	☐	☐	☐	☐

L1_1 Entwurf einer Tabelle erstellen – Lösungen

1 Definitionen

Datensatz	Mehrere Datenfelder einer Datenbanktabelle, die Informationen zu einem Datenobjekt (z.B. eine Person, ein Artikel) enthalten. Eine Zeile einer Datenbanktabelle
Primärschlüssel	Ein Attribut einer Datenbanktabelle, das jeden in dieser Tabelle gespeicherten Datensatz eindeutig identifiziert.
Attribut	Eine konkrete Eigenschaft, die einem Datenobjekt zugeordnet werden kann. Eine Spalte einer Datenbanktabelle.
Attributwert	Ein konkreter Wert, der einem Attribut eines Datensatzes zugewiesen werden kann. Schnittpunkt zwischen Zeilen und Spalten einer Datenbanktabelle.

2 Tabellenentwurf

Tabellenname: **fahrschueler**									
Attribut-name	schuelernr	nachname	vorname	telefon	email	strasse	hausnr	plz	ort
Datentyp	INT	VARCHAR	VARCHAR	VAR-CHAR	VAR-CHAR	VAR-CHAR	VAR-CHAR	VAR-CHAR	VAR-CHAR
max. Zeichen-anzahl		28	28	28	28	28	4	5	28
Primär-schlüssel	☒								

L1_2 Datenbank mit einer Tabelle modellieren – Aufgaben

Heiner Blechle zeigt seinem Freund Alexander den von Ihnen entwickelten Tabellenentwurf. Er empfiehlt ihm, den Entwurf um zwei weitere Attribute zu erweitern, um das Geburtsdatum der Fahrschüler (geburtsdatum) und die Anzahl der Fahrstunden (fahrstundenzahl) erfassen zu können.

Alexander erklärt, dass der erweiterte Entwurf nun mit Hilfe einer Software modelliert und in eine Datenbank überführt werden muss. Er schlägt vor, die Software MySQL Workbench zu verwenden. Hierbei handelt es sich um ein grafisches Entwicklungssystem zur Erstellung, Bearbeitung und Verwaltung von Datenbanken.

Beachten Sie zur Bearbeitung der folgenden Aufgabenstellungen das Informationsmaterial *L1_2.1 Information Datenbank modellieren.docx*.

1 Welche Aufgaben hat ein Entity-Relationship-Diagramm?

2 Erstellen Sie ein Entity-Relationship-Diagramm für die Daten der Fahrschüler.

3 Welche Aufgaben erfüllt ein Relationenmodell?

4 Erstellen Sie auf der Grundlage des entwickelten Entity-Relationship-Diagramms ein entsprechendes Relationenmodell.

Beachten Sie zur Bearbeitung der folgenden Aufgabenstellung das Informationsmaterial *L1_2.2 Information Datenbank softwaregestützt modellieren.docx*.

5 Erstellen Sie ein Datenmodell für die Datenbank *fahrschule* mit der Tabelle *fahrschueler* softwaregestützt mit Hilfe der MySQL Workbench.
Speichern Sie Ihr Modell unter dem Namen 'L1_2 Lösung fahrschule.mwb'.

L1_2.1 Datenbank modellieren – Information

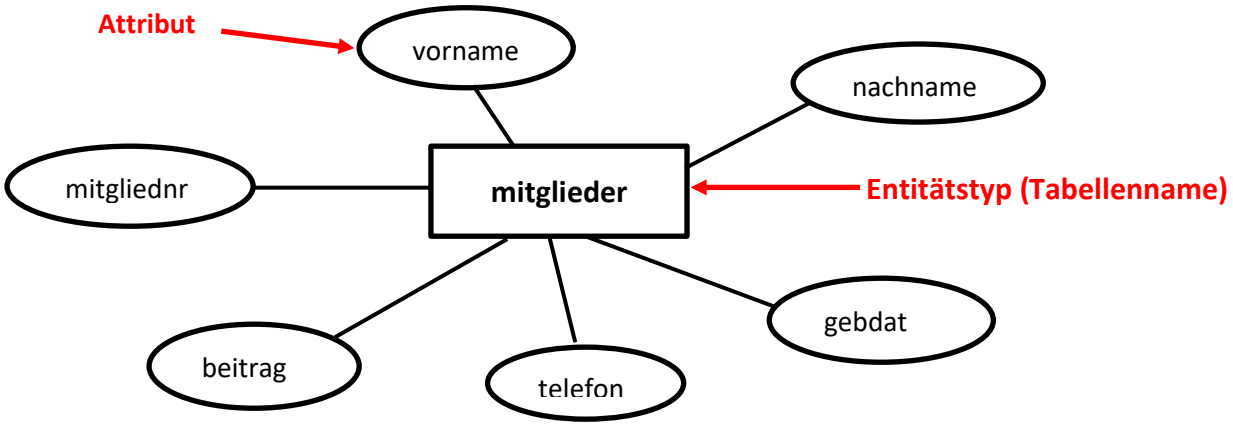
Die Datenbankmodellierung und das Arbeiten mit SQL (**Structured Query Language)** kann mit unterschiedlichen relationalen Datenbankmanagementsystemen (**RDBMS**) realisiert werden. Im Folgenden wird als RDBMS die Open-Source-Software MySQL verwendet. MySQL bietet als bevorzugtes Einsatzgebiet die Datenspeicherung für Webdienste. Sie kann deshalb nur auf einem Server gespeichert werden. Eine vorkonfigurierte Serverumgebung kann mit Hilfe der freien Software XAMPP lokal installiert werden. Man spricht dann von einem lokalen Datenbankserver.

Entity-Relationship-Diagramm

Ein Entity-Relationship-Diagramm (ERD) stellt einen Standard für die Modellierung von Datenbankbeziehungen dar. Mit Hilfe des Modells wird u.a. der logische Aufbau einer Datenbank dargestellt. Es bildet die Datenstrukturen ab und dient zum einen in der konzeptionellen Phase der Anwendungsentwicklung der Verständigung zwischen Anwendern und Entwicklern. Zum anderen ist das ER-Modell in der Implementierungsphase die Grundlage für das Design der Datenbank.

Die Modellierung beschreibt in erster Linie die benötigten Tabellen (Fachbegriff: Entitätstypen) und deren Beziehungen zueinander.

Bei einem Verein, der mit Hilfe einer Datenbank die Vor- und Nachnamen, das Geburtsdatum, die Telefonnummer und den jeweiligen Monatsbeitrag seiner Mitglieder verwalten will, ergibt die Datenmodellierung folgendes ER-Diagramm:



Auf die Darstellung der Attribute kann aus Gründen der besseren Übersicht auch verzichtet werden.

Für die Schreibweise der Entitätstypen gilt folgende Konvention:

Bezeichnung des Entitätstyps:
- Kleinbuchstabe als erstes Zeichen
- Bezeichnung im Plural.

Bezeichnung der Attribute:
- Kleinbuchstabe als erstes Zeichen

Relationenmodell erstellen

Die Modellierung eines Entity-Relationship-Diagramms dient als Grundlage für einen Datenbankentwurf und beschreibt in erster Linie die benötigten Tabellen (Entitätstypen) und deren Beziehung untereinander.

Nach der Modellierung des ER-Diagramms muss dieses Modell in ein Relationenschema (Relationenmodell) überführt werden, das in einer konkreten Datenbank implementiert werden kann. Dazu müssen für die einzelnen Tabellen (Fachbegriff: Relationen) folgende Festlegungen getroffen werden:

- Attribute
- Datentyp der Attribute
- Max. Länge der Attribute vom Typ Text
- Schlüsselattribute

Ein **Relationenmodell** beschreibt also die in einer relationalen Datenbank zu implementierenden Relationen (Tabellen), deren Attribute, die festgelegten Datentypen sowie die verwendeten Schlüssel.

Zur Darstellung eines Relationenmodells wird folgende Schreibweise verwendet:

relationenname (<u>primärschlüsselattribut DATENTYP</u>, attributname1 DATENTYP,
attributname2 DATENTYP, ….)

(Das Primärschlüsselattribut wird unterstrichen dargestellt.)

Relationenmodell am Beispiel einer Vereinsmitgliederverwaltung:

mitglieder(<u>mitgliednr INT</u>, vorname VARCHAR, nachname VARCHAR, gebdat DATE,
telefon VARCHAR, beitrag DOUBLE)

Hinweis: Aus Vereinfachungsgründen kann auf die Darstellung der Datentypen auch verzichtet werden.

Für die Schreibweise der Relationen gilt folgende Konvention:

Bezeichnung der Relation:
- Kleinbuchstabe als erstes Zeichen
- Bezeichnung im Plural.

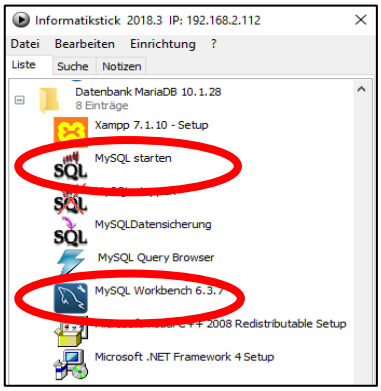
Bezeichnung der Attribute:
- Kleinbuchstabe als erstes Zeichen

L1_2.2 Datenbank softwaregestützt modellieren – Information

ER-Modell softwareunterstützt erstellen

In der Softwareentwicklung wird zunächst ein Datenbank-Modell (=Design) für eine zu entwickelnde Softwarelösung erstellt.

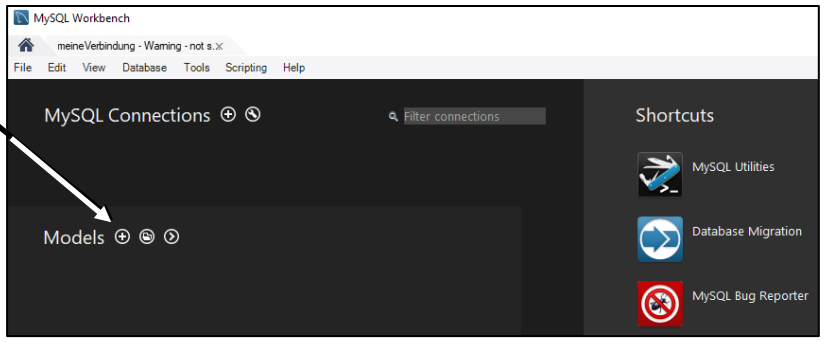
Um softwaregestützt modellieren zu können, muss die Anwendung 'MySQL Workbench' gestartet werden.



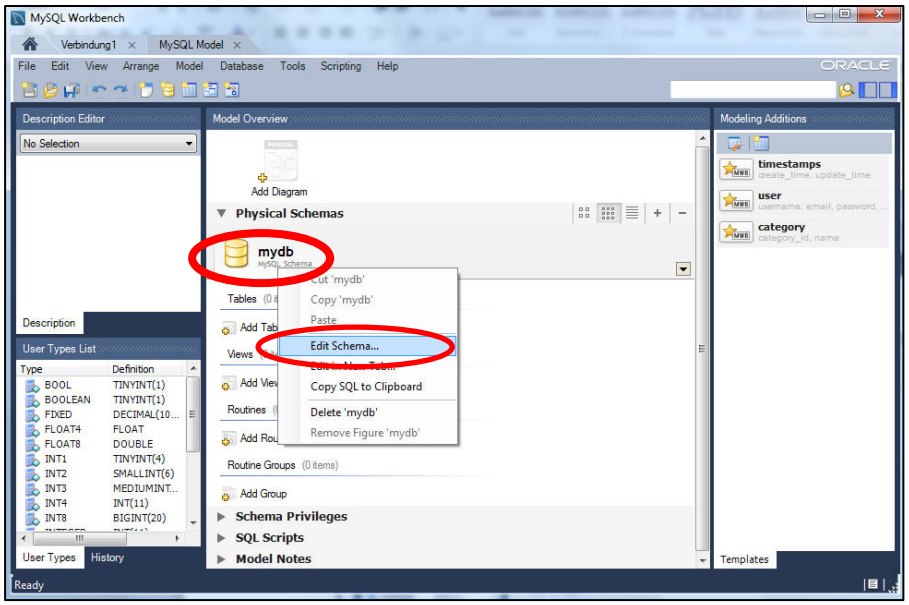
Für den späteren Zugriff auf eine Datenbank wird ein lokaler Datenbankserver benötigt, der mithilfe der Anwendung 'MySQL starten' gestartet wird

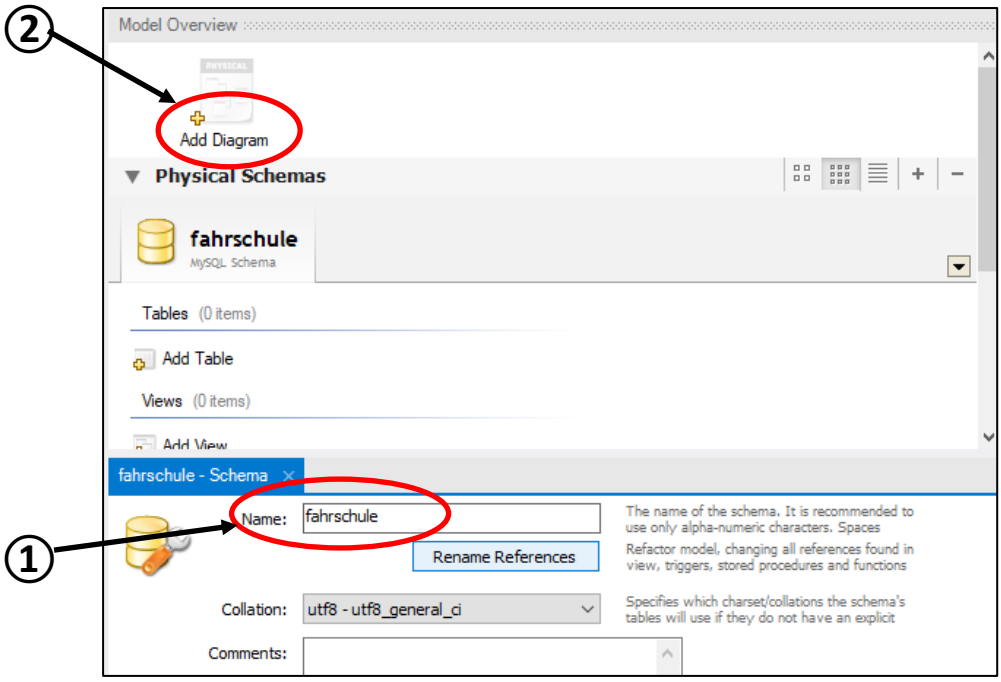
Erstellen eines neuen Datenbankmodells

- Das Symbol ⊕ öffnet das Data-Modelling-Tool der MySQL Workbench.



- Klick mit rechter Maustaste auf die bereits als Standard existierende Datenbank (= Schema) **mydb.**
- Edit Schema…

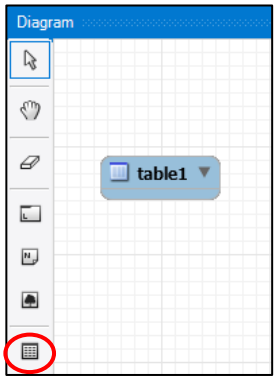




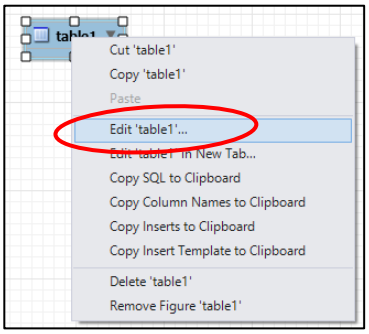
- ① Den Namen von mydb zu *fahrschule* ändern. (Kleinschreibung)

- ② Per Doppelklick auf die Schaltfläche 'Add Diagram' ein neues ER-Diagramm erstellen.
  Hinweis: Die MySQL-Workbench verwendet die Bezeichnung EER-Diagramm (erweitertes ER-Diagramm). Erläuterungen hierzu, siehe unten.

- Tabelle als Symbol wählen und auf die Zeichnungsfläche klicken.



- Klick mit rechter Maustaste auf die gezeichnete Tabelle *table1* und Auswahl des Befehls 'Edit table1…'.



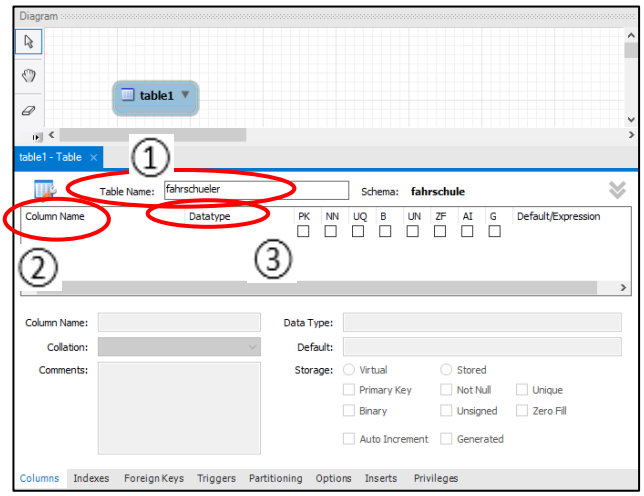
- Im sich öffnenden Dialogfenster werden

  ① der Tabellenname (Eingabe im Feld 'Table Name'),

  ② die Attributnamen (Doppelklick in eine Zelle der Spalte 'Column Name'),

  ③ die Datentypen (Auswahl in der Spalte 'Datatype')

  festgelegt.

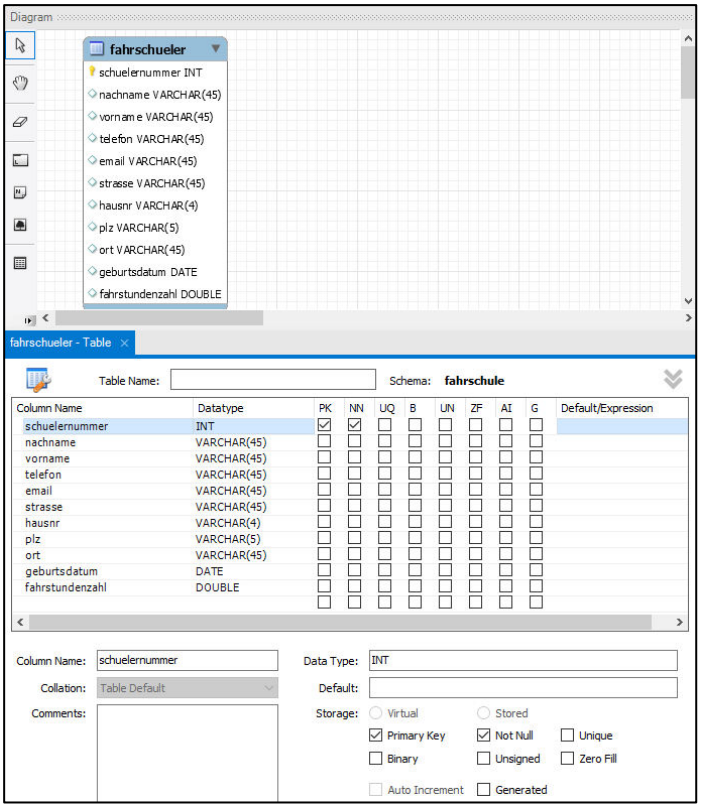


Das als erstes eingegebene Attribut wird vom System als Primärschlüssel festgelegt (PK= Primary Key).

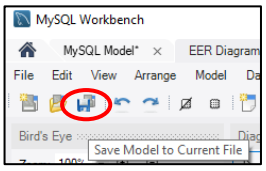
NN bedeutet Not Null, d.h. es darf nicht Nichts drinstehen.

→ das Attribut muss einen Wert enthalten.

Der Datentyp Text heißt bei MySQL VARCHAR. In der Klammer wird die max. Zeichenanzahl angegeben. Der Standardeintrag ist 45 und kann ggf. geändert werden.



Anschließend wird per Klick auf das Diskettensymbol das ER-Diagramm unter dem Namen *fahrschule_1_1.mwb* gespeichert.



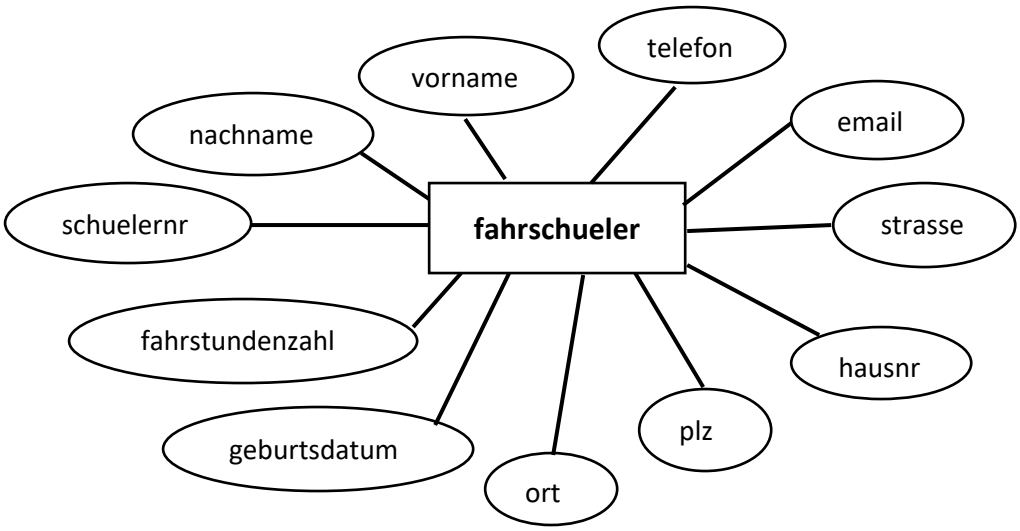
ER-Diagramm und eER-Diagramm

Das EERM der MySQL Workbench stellt sowohl die Entitäten und deren Beziehungen als auch das Relationenmodell dar und ermöglicht somit, aus dem Modell eine Datenbank zu generieren.

L1_2 Datenbank mit einer Tabelle modellieren – Lösungen

1 Ein **Entity-Relationship-Diagramm** ist die Grundlage für das Design einer Datenbank. Es beschreibt in erster Linie die benötigten Tabellen (Entitätstypen) und deren Beziehungen zueinander.

2

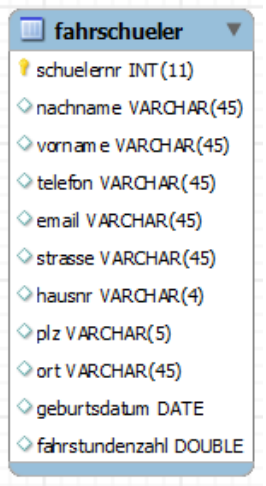


3 Ein **Relationenmodell** beschreibt die in einer relationalen Datenbank zu implementierenden Relationen (Tabellen), deren Attribute, die festgelegten Datentypen sowie die verwendeten Schlüssel.

4 fahrschueler(<u>schuelernr INT</u>, nachname VARCHAR, vorname VARCHAR, telefon VARCHAR, email VARCHAR, strasse VARCHAR, hausnr VARCHAR, plz VARCHAR, ort VARCHAR, geburtsdatum DATE, fahrstundenzahl DOUBLE)

Hinweis: Aus Vereinfachungsgründen kann auf die Darstellung der Datentypen verzichtet werden.

5 Siehe *L1_2 Lösung fahrschule.mwb.*



L1_3 Datenbank generieren – Aufgabe

Heiner Blechle zeigt seinem Freund Alexander das mit der Software MySQL Workbench erstellte Datenmodell für seine Datenbank *fahrschule*.

Alexander erklärt, dass der Entwurf nun mit der Software MySQL Workbench in eine Datenbank überführt werden kann.

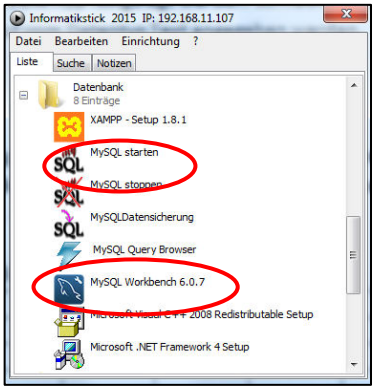
Erstellen Sie mit Hilfe der Software MySQL Workbench eine Datenbank *fahrschule* mit der Tabelle *fahrschueler*.

Beachten Sie das Informationsmaterial *L1_3 Information Datenbank generieren.docx*.

L1_3 Datenbank generieren – Information

Auf der Grundlage der Modellierung soll nun eine Datenbank generiert werden, um Daten speichern zu können.
Deshalb muss auf einem Datenbank-Server eine neue Datenbank entsprechend der Struktur des ER-Diagramms generiert werden.

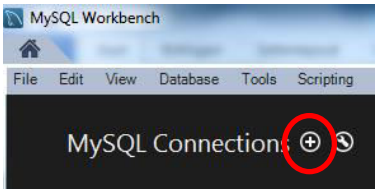
Hierfür sind die folgenden Schritte erforderlich:



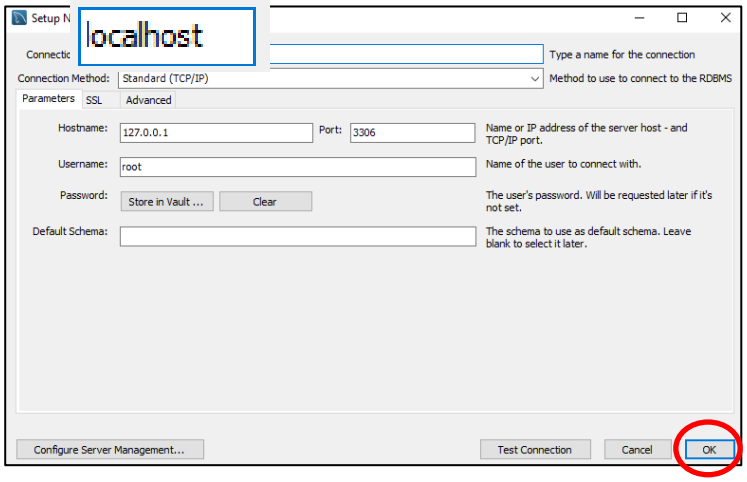
a) Starten des MySQL-Servers

b) Starten der Workbench

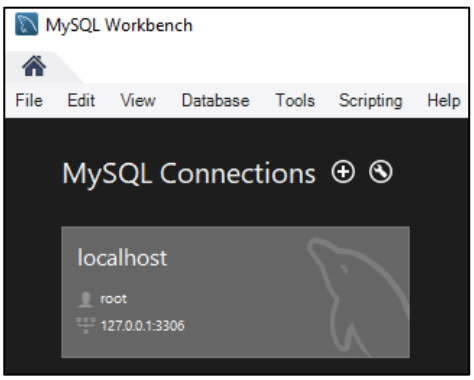
c) Erstellen einer Verbindung von der Workbench zum MySQL-Server. ⊕



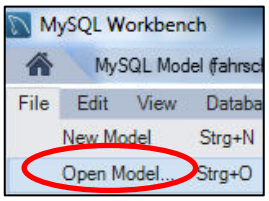
d) Eingabe eines Verbindungsnamens (z. B. localhost).



Über diese erstellte Verbindung kann später auf die Datenbank zugegriffen werden.

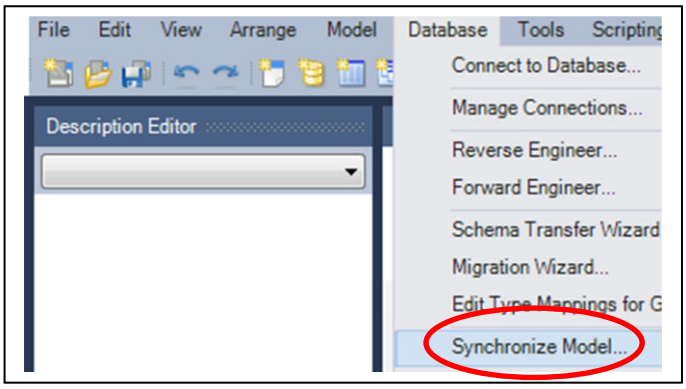


e) Das vorher gespeicherte Modell *'L1_2 Lösung fahrschule.mwb'* wird über das Menü ***File >> Open Model…*** geöffnet



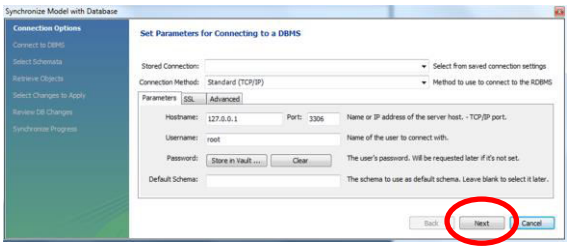
(Alternativ können Sie auch die Vorlag *L1_3 Vorlage Datenmodell fahrschule.mwb* verwenden.)

f) Mit dem Menüpunkt ***Database >> Synchronize Model..*** wird das erstellte Modell in eine Datenbank auf dem MySQL-Server umgewandelt.

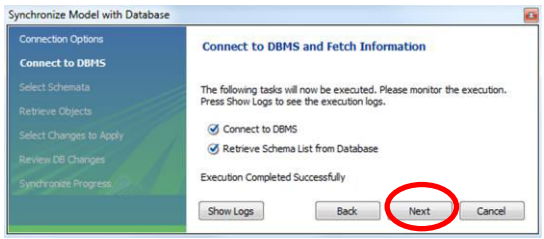


g) Die nachfolgenden Schritte werden jeweils ohne weitere Eingaben mit einem Klick auf Next bestätigt und der Vorgang im letzten Schritt mit Close abgeschlossen.

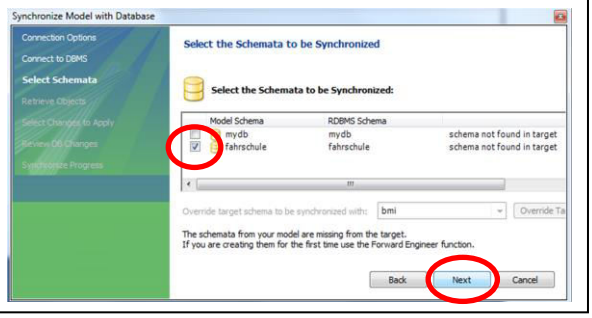
1.



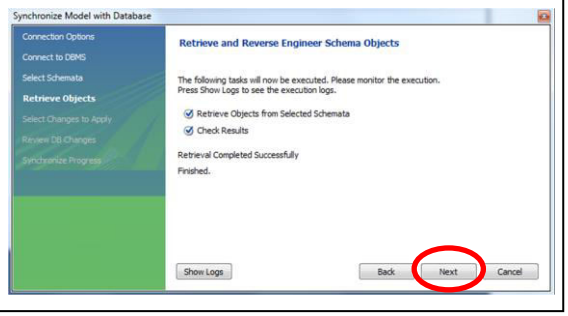
2.



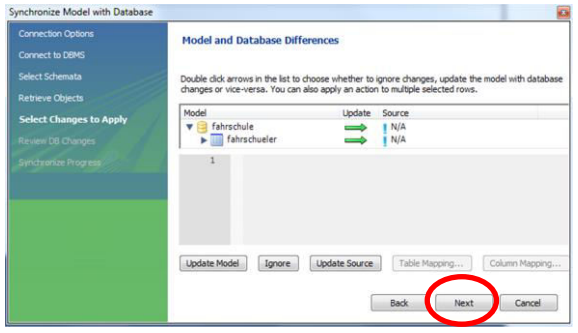
3.



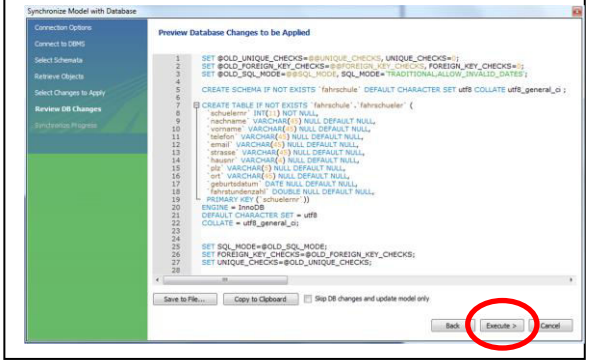
4.



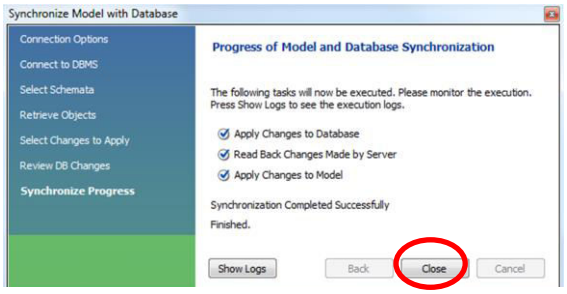
5.



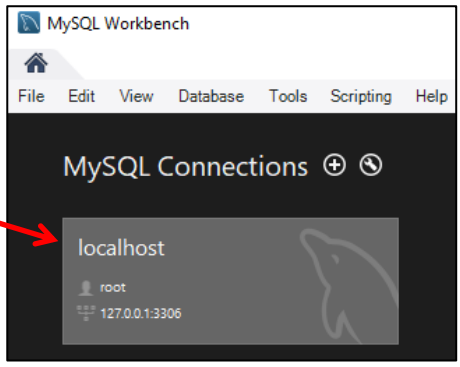
6.



7.



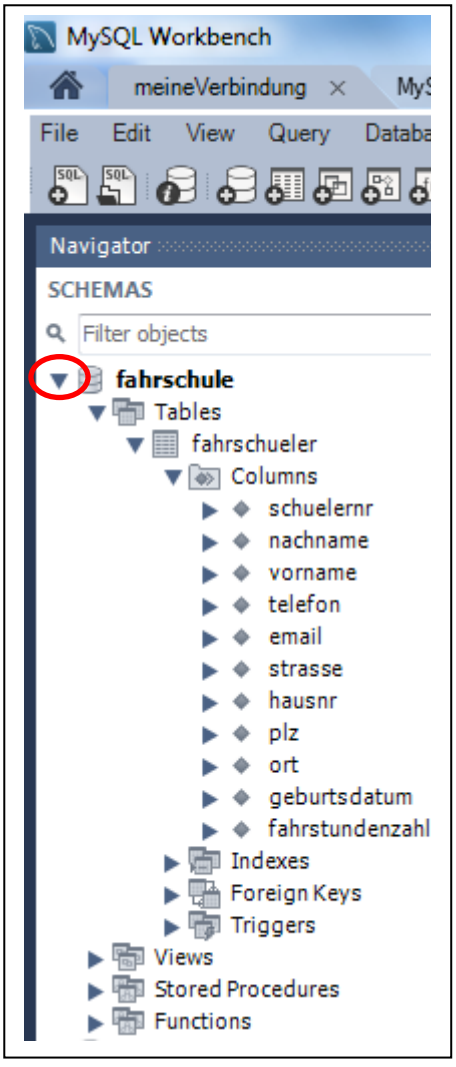
h) Öffnen der bestehenden Datenbankverbindung mit einem Klick auf 'localhost'



In der Datenbankübersicht ist nun die erstellte Datenbank fahrschule sichtbar.

Sollte dies nicht der Fall sein, so aktualisieren Sie die Ansicht per rechtem Maustastenklick → 'Refresh All'.

Die einzelnen Bestandteile der Datenbank werden durch Klick auf die Dreiecke aufgeklappt.



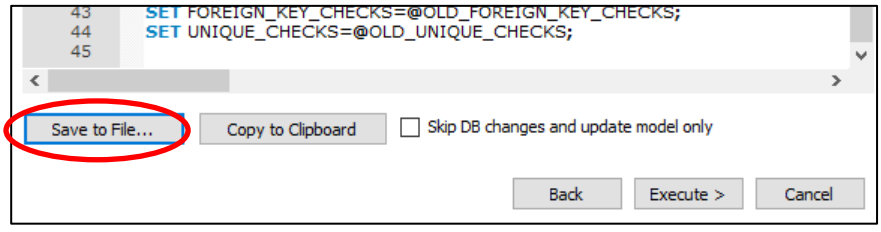
Wir werfen einen Blick hinter die Kulissen, um zu erkennen, wie die Workbench diese MySQL-Datenbank erzeugt hat:

Alle Aktionen werden in einer MySQL-Datenbank durch Befehle der Datenbanksprache SQL (Structured Query Language) ausgelöst. Erkennbar wird dies anhand der folgenden Abbildung. (entspricht Schritt 6 unter g).

```
5   CREATE SCHEMA IF NOT EXISTS `fahrschule` DEFAULT CHARACTER SET utf8 ;
6   USE `fahrschule` ;
7
8   -- -----------------------------------------------------
9   -- Table `fahrschule`.`fahrschueler`
10  -- -----------------------------------------------------
11  CREATE TABLE IF NOT EXISTS `fahrschule`.`fahrschueler` (
12    `schuelernr` INT(11) NOT NULL,
13    `nachname` VARCHAR(45) NULL DEFAULT NULL,
14    `vorname` VARCHAR(45) NULL DEFAULT NULL,
15    `telefon` VARCHAR(45) NULL DEFAULT NULL,
16    `email` VARCHAR(45) NULL DEFAULT NULL,
17    `strasse` VARCHAR(45) NULL DEFAULT NULL,
18    `hausnr` VARCHAR(4) NULL DEFAULT NULL,
19    `plz` VARCHAR(5) NULL DEFAULT NULL,
20    `ort` VARCHAR(45) NULL DEFAULT NULL,
21    `geburtsdatum` DATE NULL DEFAULT NULL,
22    `fahrstundenzahl` DOUBLE NULL DEFAULT NULL,
23    PRIMARY KEY (`schuelernr`))
24  ENGINE = InnoDB
```

Zeile	Erläuterung
5	Erstelle eine Datenbank – wenn sie nicht schon existiert – namens *fahrschule*
6	Benutze die Datenbank *fahrschule*
11	Erstelle eine Tabelle – wenn sie nicht schon existiert – namens fahrschule.*fahrschueler*.
12-22	Namen und Datentyp der Attribute. Nicht-Schlüssel-Attribute können nichts (=NULL) enthalten.
23	Der Primärschlüssel ist das Attribut *schuelernr* .

Der generierte Befehl kann auch als Skriptdatei gespeichert werden.
Damit ist es jederzeit möglich, die Datenbank in ein RDBMS zu importieren.



L1_4 Daten in eine Datenbank importieren – Aufgabe

Heiner Blechle möchte in der neuen Datenbank *fahrschueler* die Daten seiner Fahrschüler einfügen. Sein Freund Alexander hat die Kontaktdaten von Heiners Smartphone in eine Skriptdatei überführt und um die fehlenden Informationen ergänzt.

Die Datei *L1_4.1_fahrschule_1_Tabelle_Daten_einfügen.sql* steht Ihnen im Ordner '*Scripte'* in digitaler Form zur Verfügung.

Importieren Sie die Daten in die Tabelle *fahrschueler* der Datenbank *fahrschule*.

Beachten Sie das Informationsmaterial *L1_4 Information Daten importieren.docx*.

Hinweis: Sofern Sie das Datenbankmodell der Fahrschule nicht in eine Datenbank überführt haben bzw. beim Generieren der Datenbank Fehler auftraten, importieren Sie die komplette Datenbank mit der Datei *L1_4.2_fahrschule_1_Tabelle_komplett.sql.*

L1_4 Daten in eine Datenbank importieren – Information

Die bisherige Datenbanktabelle *fahrschueler* enthält noch keine Datensätze. Mit Hilfe eines vorgegebenen SQL-Scripts können Fahrschülerdaten in die Tabelle geschrieben werden.

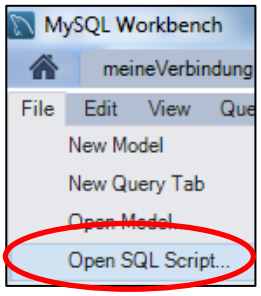
Vorgehensweise:

a) Starten des MySQL-Servers.

b) Starten der Workbench.

c) Herstellen der Verbindung von der Workbench zum MySQL-Server.

d) Mit der Befehlsfolge *File >> Open SQL Script...* die Datei *L1_4_fahrschule_1_Tabelle_Daten_einfügen.sql* öffnen



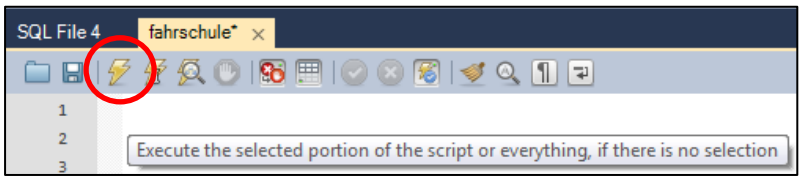
Das Script hat folgenden Inhalt (Auszug):

```
INSERT INTO `fahrschueler` VALUES
(1,'Abele','Andreas','0159787383','andreas@abele.de','Ahornweg','3','73614','Schorndorf','1999-12-02',2),
(2,'Beutel','Barbara','016673344','babsib@mail.de','Bahnhofstraße','52','73642','Welzheim','1998-06-20',12),
(3,'Cramer','Conradt','01785521221','cc@cramer.info','Cäcilienweg','34 A','73614','Schorndorf','1995-07-15',3),
(4,'Deiß','Dagmar','0166876434','d.deiss@gmx.de','Hauptstraße','44','73655','Plüderhausen','1998-11-23',14),
(5,'Emmrich','Anton','0155494521','toni.e@yahoo.com','Ginsterweg','11','73099','Adelberg','1997-03-24',6),
(6,'Dressel','Dagmar','0161745119','dagi99@web.de','Drossselweg','2','73614','Schorndorf','1999-02-04',1),
(7,'Fernandes','Enrico','01778855','fernandes.enrico@t-online.de','Konrad-Adenauer-Straße','123','73547','Lorch','1
(8,'Lutz','Frederik','016788913','lutze@gmail.com','Mozartstraße ','88','73614','Schorndorf','1998-12-24',25),
(9,'Grund','Dietmar','015545785','dgrund@web.de','Meisenweg','1','73655','Plüderhausen','1997-09-20',13),
(10,'Demürel','Ali','01632382','alibaba99@yahoo.com','Webergasse','14','73642','Welzheim','1998-12-23',23),
(11,'Dressel','Andreas','015526372','andidressel@gmail.com','Drosselweg','4','73614','Schorndorf','1997-10-22',3),
(12,'Schmidt','Alex','01563723','aschmidt@yahoo.com','Bahnhofstraße','34','73642','Welzheim','1998-07-30',22);
```

Erläuterung:

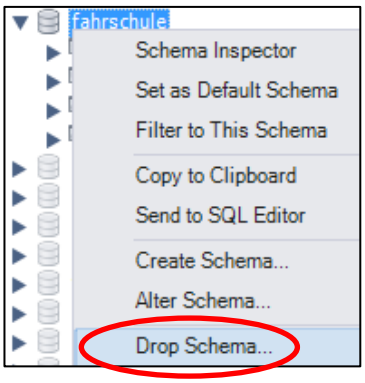
INSERT INTO `fahrschueler` VALUES (1,'Abele','Andreas','0159787383','andreas@abele.de','Ahornweg','3','73614', 'Schorndorf', '1999-12-02',2), ….
Füge einen Datensatz in die Tabelle 'fahrschueler' ein mit den Werten (1,'Abele','Andreas','0159787383','andreas@abele.de','Ahornweg','3','73614', 'Schorndorf', '1999-12-02',2), ….

e) Das Script ausführen durch einen Klick auf den „Blitz“

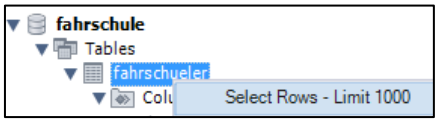


Hinweis:

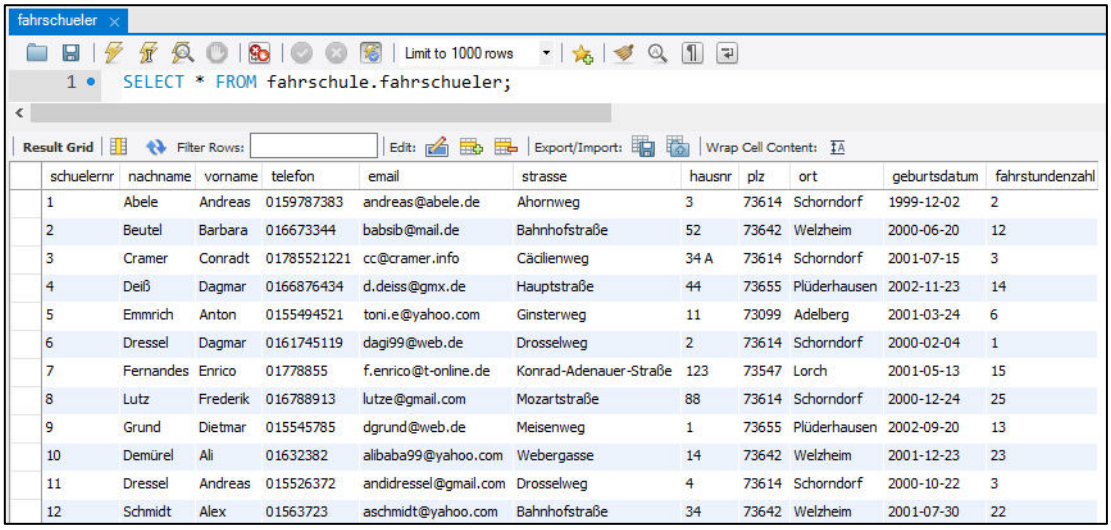
Für den Fall, dass bei Aufgabe e) Fehler auftreten, kann mit Hilfe des Skripts *L1_4_fahrschule_1_Tabelle_komplett.sql* die vollständige Datenbank mit allen erforderlichen Tabellen und Daten erzeugt werden. Zuvor muss die bisherige Datenbank *fahrschule* gelöscht werden.



f) Der Inhalt der Tabelle kann über das Kontextmenü zur Tabelle *fahrschueler* (rechte Maus) angezeigt werden.



Durch den so erzeugten SQL-Befehl werden alle Datensätze ausgewählt und angezeigt.

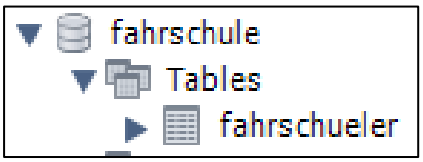


schuelernr	nachname	vorname	telefon	email	strasse	hausnr	plz	ort	geburtsdatum	fahrstundenzahl
1	Abele	Andreas	0159787383	andreas@abele.de	Ahornweg	3	73614	Schorndorf	1999-12-02	2
2	Beutel	Barbara	016673344	babsib@mail.de	Bahnhofstraße	52	73642	Welzheim	2000-06-20	12
3	Cramer	Conradt	01785521221	cc@cramer.info	Cäcilienweg	34 A	73614	Schorndorf	2001-07-15	3
4	Deiß	Dagmar	0166876434	d.deiss@gmx.de	Hauptstraße	44	73655	Plüderhausen	2002-11-23	14
5	Emmrich	Anton	0155494521	toni.e@yahoo.com	Ginsterweg	11	73099	Adelberg	2001-03-24	6
6	Dressel	Dagmar	0161745119	dagi99@web.de	Drosselweg	2	73614	Schorndorf	2000-02-04	1
7	Fernandes	Enrico	01778855	f.enrico@t-online.de	Konrad-Adenauer-Straße	123	73547	Lorch	2001-05-13	15
8	Lutz	Frederik	016788913	lutze@gmail.com	Mozartstraße	88	73614	Schorndorf	2000-12-24	25
9	Grund	Dietmar	015545785	dgrund@web.de	Meisenweg	1	73655	Plüderhausen	2002-09-20	13
10	Demürel	Ali	01632382	alibaba99@yahoo.com	Webergasse	14	73642	Welzheim	2001-12-23	23
11	Dressel	Andreas	015526372	andidressel@gmail.com	Drosselweg	4	73614	Schorndorf	2000-10-22	3
12	Schmidt	Alex	01563723	aschmidt@yahoo.com	Bahnhofstraße	34	73642	Welzheim	2001-07-30	22

SELECT * FROM fahrschule.fahrschueler;
Wähle alle Attributinhalte von der Tabelle *fahrschueler* aus der Datenbank *fahrschule*

* bedeutet alle Attribute (Spalten)

Per Doppelklick auf den Namen der Datenbank *fahrschule* wird diese zur aktuellen Datenbank erklärt – man erkennt dies daran, dass der Name nun fett geschrieben erscheint.



In der Folge kann man den o.g. SQL-Befehl auch folgendermaßen schreiben – nämlich ohne Nennung der Datenbank:

```
SELECT *
FROM fahrschueler;
```

Ein SQL-Befehl wird grundsätzlich mit einem Semikolon abgeschlossen.

L1_5.1 Daten einer Datenbank mit SQL abfragen. Projektion – Aufgaben

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informationsmaterial *L1_5.1 Information Datenbankabfrage Projektion.docx.*

1 Erläutern Sie den Begriff 'Projektion'.

2 Heiner Blechle möchte aus seiner Datenbank *fahrschule* die Informationen Schülernummer sowie Vorname und Nachname aufgelistet haben.
Die Liste soll nach den Nachnamen alphabetisch geordnet sein.

Formulieren Sie einen entsprechenden SQL-Befehl.

Dokumentieren Sie Ihre Lösung.

3 Alle Fahrschüler mit Schülernummer, Vor- und Nachnamen sowie Wohnort sollen nach Wohnort absteigend geordnet (sortiert) aufgelistet werden.

Formulieren Sie einen entsprechenden SQL-Befehl.

Dokumentieren Sie Ihre Lösung.

4 Alle Fahrschüler mit Schülernummer sowie Vor- und Nachnamen sollen aufgelistet werden. Die Liste soll alphabetisch nach den Nachname geordnet (sortiert) sein. Bei gleichen Nachnamen soll nach den Vornamen sortiert werden.

Formulieren Sie einen entsprechenden SQL-Befehl.

Dokumentieren Sie Ihre Lösung.

L1_5.1 Daten einer Datenbank mit SQL abfragen, Projektion – Information

In den seltensten Fällen werden alle Daten einer Datenbanktabelle benötigt. Die Datenbanksprache SQL stellt hierzu den SELECT-Befehl zur Verfügung.

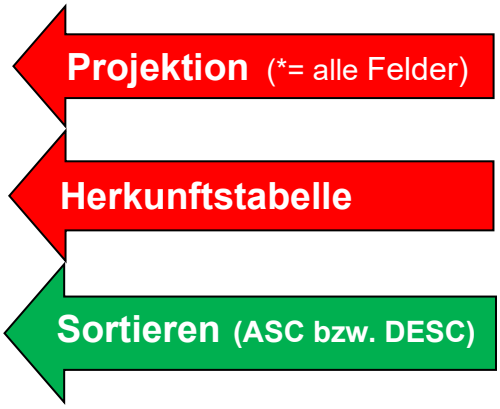
Sollen beispielsweise nicht alle Attribute einer Tabelle aufgelistet werden, spricht man von einer **Projektion.**

Befehlssyntax der Auswahlanweisung SELECT

SELECT *attributname1*, *attributname2*…

FROM *tabellenname*

[**ORDER BY** *attributname1*, *attributname2*, … [ASC|DESC]]



Hinweis: SQL unterscheidet nicht zwischen Groß- und Kleinschreibung. Zur besseren Übersicht empfiehlt es sich, die SQL-Schlüsselworte in Großbuchstaben zu schreiben.

Übersicht zu den verwendeten SQL - Klauseln

Die **SELECT** - Klausel	Hinter **SELECT** stehen die Namen der Attribute, deren Inhalte aufgelistet werden sollen. Die Verwendung des Zeichens * führt dazu, dass die Werte aller Attribute aufgelistet werden.
Die **FROM** - Klausel	Hinter **FROM** steht der Name der Tabelle, aus der die Attribute stammen.
Die **ORDER BY** - Klausel	Daten werden nach einem oder mehreren Attributwerten **sortiert** ausgegeben. Die **vorgegebene Sortierreihenfolge ist aufsteigend *ASC*** (muss nicht angeben werden). Bei mehreren Sortierkriterien (Nachname, Vorname) müssen die Attributnamen mit Komma getrennt genannt werden. Soll absteigend sortiert werden, muss ***DESC*** eingegeben werden.

Beispiele für SQL-Befehle zur Auswahl und Sortierung bestimmter Attribute (Spalten):

1. Die Adressen aller Fahrschüler sollen in alphabetischer Reihenfolge der Ortsnamen aufgelistet werden:

   ```
   SELECT strasse, hausnr, plz, ort
   FROM fahrschueler
   ORDER BY ort;
   ```

2. Die Auflistung soll nach Postleitzahlen absteigend sortiert sein. Bei gleichen Postleitzahlen soll die Sortierung nach den Ortsnamen aufsteigend geordnet sein:

   ```
   SELECT strasse, hausnr, plz, ort
   FROM fahrschueler
   ORDER BY plz DESC, ort;
   ```

Zur Dokumentation Ihrer Ergebnisse können Sie Ihre Eingabe in der MySQL-Workbench in die Zwischenablage kopieren und in ein Worddokument einfügen.

L1_5.1 Daten einer Datenbank abfragen, Projektion – Lösungen

1 Eine Projektion wählt bestimmte Attribute aus der Gesamtmenge aller Attribute aus.

2
```
SELECT schuelernr, vorname, nachname
FROM fahrschueler
ORDER BY nachname;
```

3
```
SELECT schuelernr, vorname, nachname, ort
FROM fahrschueler
ORDER BY ort DESC;
```

4
```
SELECT schuelernr, vorname, nachname
FROM fahrschueler
ORDER BY nachname, vorname;
```

L1_5.3 Daten einer Datenbank mit SQL abfragen. Selektion – Aufgaben

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informationsmaterial *L1_5.3 Information Datenbankabfrage Selektion.docx.*

1 Erläutern Sie den Begriff 'Selektion'.

2 Heiner Blechle möchte aus seiner Datenbank alle Informationen zu seinen Fahrschülern aus Schorndorf aufgelistet haben.

Formulieren Sie einen entsprechenden SQL-Befehl und dokumentieren Sie Ihre Lösung.

Formulieren Sie für die folgenden Aufgabenstellungen die entsprechenden SQL-Befehle und dokumentieren Sie Ihre Lösungen.

3 Alle Fahrschüler mit Vor- und Nachnamen sowie deren Geburtsdatum, die mit Nachnamen Dressel heißen, sollen aufgelistet werden.

4 Alle Fahrschüler mit Vor- und Nachnamen sowie deren Geburtsdatum, die mit Nachnamen Dressel heißen, sollen aufgelistet werden. Die Auflistung soll alphabetisch nach den Vornamen geordnet sein.

5 Alle Fahrschüler, die mehr als 20 Fahrstunden haben, sollen mit allen Informationen aufgelistet werden.

6 Alle Fahrschüler, die vor 2001 geboren sind, sollen mit allen Informationen aufgelistet werden.

7 Alle Fahrschüler, die vor 2001 geboren sind, sollen nach dem Geburtsdatum absteigend mit allen Informationen aufgelistet werden.

8 Alle Fahrschüler, deren Nachname mit „D“ anfängt, sollen mit allen Informationen aufgelistet werden.

9 Alle Fahrschüler aus Schorndorf, die im Drosselweg wohnen, sollen mit allen Informationen aufgelistet werden.

10 Alle Fahrschüler, die im Zeitraum vom 1. Januar 2000 bis zum 31. Dezember 2001 geboren sind, sollen mit allen Informationen aufgelistet werden.

11 Alle Fahrschüler, die vor dem 01.01.2000 oder die nach dem 31.12.2001 geboren sind, sollen mit allen Informationen aufgelistet werden.

12 Alle Fahrschüler, die nicht in Schorndorf wohnen, sollen mit allen Informationen aufgelistet werden.

L1_5.3 Daten einer Datenbanktabelle mit SQL abfragen, Selektion – Information

Sollen bei der Auswahl der Daten einer Datenbank nicht alle Datensätze (Zeilen) aufgelistet werden, spricht man von einer **Selektion**.

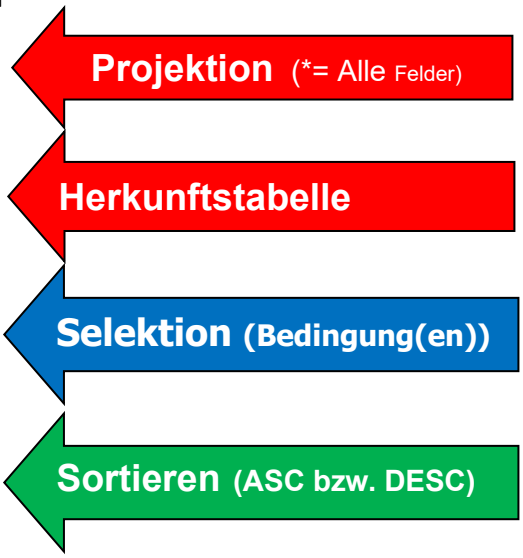
Befehlssyntax der Auswahlanweisung SELECT

SELECT *attributname1*, *attributname2* …

FROM *tabellenname*

[**WHERE** *bedingung*]

[**ORDER BY** *attributname1*, … [ASC|DESC]..]



Übersicht zur weiteren SQL - Klauseln

Die **WHERE** - Klausel	Damit wird bestimmt, welche Datensätze (Zeilen einer Tabelle) ausgewählt werden sollen. (**Bedingungsprüfung**). Enthält die Bedingungsprüfung mehrere Bedingungen, werden diese mit den logischen Operatoren AND, OR und NOT verknüpft.

Aufbau einer Bedingung:

Jede Bedingung besteht aus drei Teilen:

- Attributname, dessen Wert verglichen werden soll - z.B.: nachname
- Vergleichsoperator - z.B.: gleich (=)
- Vergleichswert - z.B.: Huber

➔ WHERE nachname = 'Huber'

Hinweis: Zu vergleichende **Texte müssen in Anführungszeichen bzw. Hochkomma** geschrieben werden.

Übersicht Operatoren

Operator	Name	Bedeutung
=	gleich	wahr, wenn die zu vergleichenden Werte identisch sind
!= oder	ungleich	wahr, wenn die zu vergleichenden Werte verschieden sind
<	kleiner als	wahr, wenn der links stehende Vergleichswert kleiner ist
>	größer als	wahr, wenn der links stehende Vergleichswert größer ist
<=	kleiner gleich	wahr, wenn der links stehende Vergleichswert kleiner oder gleich ist
>=	größer gleich	wahr, wenn der links stehende Vergleichswert größer oder gleich ist
BETWEEN	zwischen	wahr, wenn der links stehende Vergleichswert im angegebenen Wertebereich liegt (siehe unten)

Weitere Regeln zur Selektion:

- Zu vergleichende **Zahlen** müssen **ohne Anführungszeichen** geschrieben werden.
  z.B.: WHERE umsatz < 10000

- Ein zu vergleichendes **Datum** muss **mit Anführungszeichen** im **Format 'jjjj-mm-tt'** (in Anführungszeichen bzw. Hochkomma) geschrieben werden.
  z.B.: WHERE kaufdatum < '2019-08-21'

- In vielen Fällen soll nach Textmustern gesucht werden, die eine bestimmte Zeichenkette enthalten. Hierzu kann der **Operator LIKE** verwendet werden, der zwei Platzhalter zur Verfügung stellt:
  → % - für beliebig viele Zeichen z.B.: WHERE plz LIKE '70%'
  (alle Orte, deren Postleitzahl mit den Ziffern 70 beginnt.)
  → _ - für ein bestimmtes Zeichen z.B.: WHERE nachname LIKE 'M__er'
  (alle Personen die Maier, Meier oder Mayer heißen.)

- Mit Hilfe des Operators **NOT** kann überprüft werden, ob eine Bedingung nicht erfüllt ist. Er wird einer Bedingung vorangestellt und kehrt das Ergebnis der Bedingung um. Aus true wird false und aus false wird true.
  z.B.: Personen, die nicht aus Stuttgart kommen: WHERE NOT ort = 'Stuttgart'

- Eine WHERE – Klausel kann aus mehreren Bedingungsteilen bestehen. Sie werden entsprechend der Aufgabenstellung mit **AND** (bzw. &&) oder **OR** (bzw. ||) verknüpft.
  → AND - sowohl die erste als auch die zweite Bedingung muss erfüllt sein
  z.B.: WHERE ort = 'Stuttgart'
  AND nachname = 'Huber'
  (alle Personen, die Huber heißen und aus Stuttgart kommen.)

→ OR - entweder die erste oder die zweite Bedingung muss erfüllt sein
z.B.: WHERE plz = '78150'
OR plz = '79599'
(Alle die in einem Ort mit der Postleitzahl 78150 oder mit der Postleitzahl 79599 wohnen.)

→ NOT - entweder die erste oder die zweite Bedingung muss erfüllt sein
z.B.: WHERE plz = '78150'
OR plz = '79599'

- Der Operator BETWEEN prüft, ob der Attributwert innerhalb einer bestimmten Grenze liegt. Die Überprüfung kann sich auf Zahlen- oder Datumswerte beziehen.
  z.B.: WHERE mietpreis BETWEEN 500 AND 700
  WHERE kaufdatum BETWEEN '2018-01-01' AND '2089-12321'

  Zu beachten ist, dass die angegebenen Grenzwerte des Wertebereichs in die Überprüfung der Bedingung mit eingeschlossen werden.

  Am Beispiel der Bedingung "mietpreis BETWEEN 500 AND 700" bedeutet dies, dass ein Mietpreis von 500 die Bedingung erfüllt.

L1_5.3 Daten einer Datenbank abfragen, Selektion – Lösungen

1 Eine Selektion wählt bestimmte Datensätze aus der Gesamtmenge aller Datensätze aus.

2 Alle Fahrschüler aus Schorndorf.

```
SELECT *
FROM fahrschueler
WHERE ort = 'Schorndorf';
```

Hinweis: Zu vergleichende **Texte müssen in Hochkommata** geschrieben werden!

3 Alle Fahrschüler mit Vor- und Nachnamen sowie deren Geburtsdatum, die mit Nachnamen Dressel heißen.

```
SELECT vorname, nachname, geburtsdatum
FROM fahrschueler
WHERE nachname = 'Dressel';
```

4 … wie Aufgabe 2, zusätzlich nach dem Vornamen sortiert.

```
SELECT vorname, nachname, geburtsdatum
FROM fahrschueler
WHERE nachname = 'Dressel'
ORDER BY vorname;
```

5 Alle Fahrschüler, die mehr als 20 Fahrstunden haben.

```
SELECT *
FROM fahrschueler
WHERE fahrstundenzahl > 20;
```

Hinweis: zu vergleichende **Zahl ohne Hochkommata** schreiben!

6 Alle Fahrschüler, die vor 2001 geboren sind.

```
SELECT *
FROM fahrschueler
WHERE geburtsdatum < '2001-01-01';
```

Hinweis: zu vergleichendes **Datum mit Hochkommata im Format 'jjjj-mm-tt'** schreiben!

7 wie Aufgabe 5, zusätzlich nach dem Alter absteigend sortiert.

```
SELECT *
FROM fahrschueler
WHERE geburtsdatum < '2001-01-01'
ORDER BY geburtsdatum DESC;
```

8 Alle Fahrschüler, deren Nachname mit „D“ anfängt.

```
SELECT *
FROM fahrschueler
WHERE nachname LIKE 'D%';
```

Hinweis: In vielen Fällen soll nach Textmustern gesucht werden, die eine bestimmte Zeichenkette enthalten. Hierzu kann der **Operator LIKE** verwendet werden, der zwei Platzhalter zur Verfügung stellt:

→ % - für beliebig viele Zeichen

→ _ - für ein bestimmtes Zeichen

9 Alle Fahrschüler aus Schorndorf, die im Drosselweg wohnen.

```
SELECT *
FROM fahrschueler
WHERE ort = 'Schorndorf'
AND strasse = 'Drosselweg';
```

Hinweis: Die WHERE – Klausel kann aus mehreren Bedingungsteilen bestehen. Sie werden entsprechend der Aufgabenstellung mit **AND** (bzw. &&) oder **OR** (bzw. ||) verknüpft.

10 Alle Fahrschüler die im Zeitraum vom 1. Januar 2000 bis zum 31. Dezember 2001 geboren sind.

```
SELECT *
FROM fahrschueler
WHERE geburtsdatum >= '2000-01-01'
AND geburtsdatum <= '2001-12-31';
```

Alternativ:

```
SELECT *
FROM fahrschueler
WHERE geburtsdatum BETWEEN '2000-01-01' AND
                           '2001-12-31';
```

11 Eine Liste mit allen Informationen, die alle Fahrschüler enthält, die vor dem 01.01.2000 geboren sind, als auch solche, die nach dem 31.12.2001 geboren sind.

```
SELECT *
FROM fahrschueler
WHERE geburtsdatum < '2000-01-01'
OR geburtsdatum > '2001-12-31';
```

12 Alle Fahrschüler, die nicht in Schorndorf wohnen.

```
SELECT *
FROM fahrschueler
WHERE  ort != 'Schorndorf';
```

Alternativ:

```
SELECT *
FROM fahrschueler
WHERE NOT ort = 'Schorndorf';
```

ort	strasse
Schorndorf	Ahornweg
Welzheim	Bahnhofstraße
Schorndorf	Cäcilienweg
Plüderhausen	Hauptstraße
Adelberg	Ginsterweg
Schorndorf	Drosselweg
Lorch	Konrad-Adenauer-Straße
Schorndorf	Mozartstraße
Plüderhausen	Meisenweg
Welzheim	Webergasse
Schorndorf	Drosselweg
Welzheim	Bahnhofstraße

L1_5.5 Daten einer Datenbank mit SQL abfragen. Funktionen – Aufgaben

Heiner Blechle möchte mit Hilfe seiner Datenbank verschiedene Auswertungen der Fahrschülerdaten vornehmen.

Formulieren Sie für die folgenden Aufgabenstellungen die entsprechenden SQL-Befehle und dokumentieren Sie Ihre Lösungen.

Beachten Sie das Informationsmaterial
L1_5.5 Information Datenbankabfrage Funktionen.docx.

1 Wie hoch ist die Anzahl aller Fahrschüler?

2 Wie hoch ist die höchste Anzahl an Fahrstunden?

3 Wie hoch ist die durchschnittliche Anzahl an Fahrstunden?
Das Ergebnis soll mit zwei Dezimalstellen angezeigt werden.

4 Wie viele Fahrstunden wurden insgesamt durchgeführt?

5 Wie hoch ist der jeweilige Umsatz auf Grund der Fahrstunden, wenn pro Fahrstunde mit 30 € gerechnet wird?

6 Wie hoch ist der gesamte Umsatz auf Grund der Fahrstunden, wenn pro Fahrstunde mit 30 € gerechnet wird

L1_5.5 Daten einer Datenbank mit SQL abfragen. Funktionen und berechnete Felder – Information

Die Datenbanksprache SQL bietet auch Funktionen an, mit deren Hilfe Rechenoperationen auf selektierte Daten ausgeführt werden können.

Aggregatfunktionen

Aggregatfunktionen dienen zur Zusammenfassung und Verdichtung von Rechenergebnissen. Sie arbeiten über alle Zeilen der entsprechenden Tabelle bzw. der selektierten Untergruppe. Mit ihnen werden Rechenergebnisse ermittelt, bei denen genau ein Ergebniswert aus einer Vielzahl von Eingabewerten gebildet wird.

Diese Funktionen können verwendet werden, um

• die Anzahl der selektierten Datensätze zu zählen	COUNT(PrimaryKey) (oder *)
• den Maximalwert einer Spalte zu bestimmen	MAX(Attributname)
• den Minimalwert einer Spalte zu bestimmen	MIN(Attributname)
• den Durchschnittswert einer Spalte zu bestimmen	AVG(Attributname)
• die Werte einer Spalte zu addieren	SUM(Attributname)

- **COUNT()-Funktion**

  Die SQL COUNT-Funktion zählt (count) die Anzahl von ausgewählten Datensätzen. Für diese Aggregatfunktion muss das Schlüsselwort COUNT sowie das Attribut der zu zählenden Datensätze in den Klammern angegeben werden.

  Es werden alle Datensätze gezählt, deren Wert nicht NULL ist. Es empfiehlt sich deshalb das Primärschlüsselattribut zu verwenden.

  Damit die Ausgabe des Ergebnisses eine Spaltenüberschrift erhält, kann der SELECT-Anweisung ein sogenannter Alias hinzugefügt werden. Enthält der Alias Leerzeichen, so muss er in Hochkomma gesetzt werden.

  z.B.: SELECT COUNT(schuelernr) AS Anzahl_Fahrschueler
  FROM fahrschueler;

- **MAX()-Funktion**

  Die SQL MAX-Funktion ermittelt den höchsten Wert einer Tabellenspalte. Für diese Aggregatfunktion muss das Schlüsselwort MAX sowie das Attribut, dessen Höchstwert ermittelt werden soll, in den Klammern angegeben werden.

  z.B.: SELECT MAX(fahrstundenanzahl) AS hoechste_Stundenzahl
  FROM fahrschueler;

- **MIN()-Funktion**

  Die SQL MIN-Funktion ermittelt den niedrigsten Wert einer Tabellenspalte. Für diese Aggregatfunktion muss das Schlüsselwort MIN sowie das Attribut, dessen niedrigster Wert ermittelt werden soll, in den Klammern angegeben werden.

  z.B.: SELECT MIN(fahrstundenanzahl) AS geringste_Stundenzahl
  FROM fahrschueler;

- **AVG()-Funktion**

  Die SQL AVG-Funktion ermittelt den Durchschnittswert aller Werte einer Tabellenspalte. Für diese Aggregatfunktion muss das Schlüsselwort AVG sowie das Attribut, dessen Durchschnittswert ermittelt werden soll, in den Klammern angegeben werden.

  z.B.: SELECT AVG(fahrstundenanzahl) AS durchschnittliche_Stundenzahl
  FROM fahrschueler;

- **SUM()-Funktion**

  Die SQL SUM-Funktion ermittelt die Summe aller Werte einer Tabellenspalte. Für diese Aggregatfunktion muss das Schlüsselwort SUM sowie das Attribut, dessen Summe ermittelt werden soll, in den Klammern angegeben werden.

  z.B.: SELECT SUM(fahrstundenanzahl) AS Gesamtstundenzahl
  FROM fahrschueler;

Berechnete Felder

Mit Hilfe der Datenbanksprache SQL können auch Befehle formuliert werden, die **Berechnungen** enthalten.
Da die Berechnung eine virtuelle Spalte mit den berechneten Werten erzeugt, muss die entsprechende Anweisung als Projektion (hinter SELECT) geschrieben werden.

z.B.: SELECT nachname, fahrstundenzahl, fahrstundenzahl * 50 AS Umsatz
FROM fahrschueler;

Hinweis: Berechnungen können auch Funktionen enthalten.

Um das Ergebnis einer Berechnung oder einer Funktion in einem gewünschten **Zahlenformat** zu erhalten, muss die **FORMAT-Funktion** verwendet werden.

FORMAT(zahl, dezimalstellen)

Der Ausdruck

FORMAT(17.896578101,2) ergibt das Ergebnis: 17.90

Die FORMAT-Funktion kann auch in Verbindung mit anderen Funktionen oder Rechenoperationen verwendet werden.

z.B.: SELECT FORMAT(AVG(fahrstundenzahl),2)
FROM fahrschueler;

L1_5.5 Daten einer Datenbank abfragen, Funktionen – Lösungen

1 SELECT COUNT(*)
FROM fahrschueler;

Es ist sinnvoll, für das Funktionsergebnis einen Alias (Andersnamen) zu verwenden. Enthält der Alias mehrere Wörter, so müssen diese jeweils mit einem Unterstrich verbunden werden.

SELECT COUNT(*) AS Anzahl_der_Fahrschüler
FROM fahrschueler;

2 SELECT MAX(fahrstundenzahl) AS Höchste_Anzahl_an_Fahrstunden
FROM fahrschueler;

3 SELECT AVG(fahrstundenzahl) AS Durchschnittliche_Anzahl_an_Fahrstunden
FROM fahrschueler;

Soll das Ergebnis mit 2 Dezimalstellen angezeigt werden, muss die Funktion FORMAT(Ausdruck, Dezimalstellen) verwendet werden.

SELECT FORMAT(AVG(fahrstundenzahl),2) AS Durchschnittliche_Anzahl_an_Fahrstunden
FROM fahrschueler;

4 SELECT SUM(fahrstundenzahl) AS Summe_der_Fahrstunden
FROM fahrschueler;

5 SELECT nachname, vorname, fahrstundenzahl * 30 AS Umsatz_aus_Fahrstunden_in_€
FROM fahrschueler;

6 SELECT SUM(fahrstundenzahl) * 30 AS Umsatzsumme_aus_Fahrstunden_in_€
FROM fahrschueler;

L1_5.7 Redundanzen in Abfrageergebnissen vermeiden – Aufgaben

Heiner Blechle möchte weitere Auswertungen seiner Datenbank vornehmen.

Formulieren Sie dazu für die folgenden Aufgabenstellungen die entsprechenden SQL-Befehle und dokumentieren Sie Ihre Lösungen.

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informationsmaterial *L1_5.7 Information Redundanzen in Abfrageergebnissen.docx*.

1 Gewünscht wird eine Liste aller Orte mit Postleitzahl, aus denen die Schüler der Fahrschule stammen.
Gleiche Orte sollen nur einmal angezeigt werden.

2 Es sollen die Vornamen der Fahrschüler angezeigt werden. Jeder Vorname soll nur einmal angezeigt werden.

3 Es sollen die Nachnamen der Fahrschüler angezeigt werden. Jeder Nachname soll nur einmal angezeigt werden.

4 Es sollen die Fahrstundenzahlen angezeigt werden, die die Fahrschüler bisher erhalten haben. Jeder Fahrstundenzahl soll nur einmal angezeigt werden.

L1_5.7 Redundanzen in Abfrageergebnissen vermeiden – Information

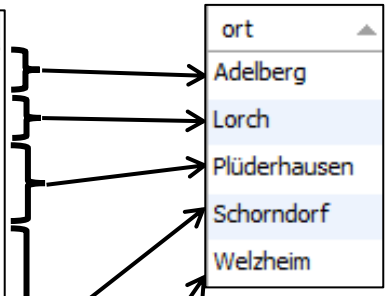
Mit der SELECT-Anweisung wird der gesamte Inhalt einer oder mehrerer Tabellenspalten einer Datenbanktabelle bzw. der selektierten Untergruppe ausgewählt und angezeigt. Dies kann dazu führen, dass bestimmte Werte mehrfach aufgelistet werden. Das Abfrageergebnis weist somit Redundanzen auf.

Am Beispiel der Auflistung aller Orte, aus denen die Fahrschüler von Heiner Blechle stammen, führt die Anweisung

```
SELECT ort, plz
FROM fahrschueler;
```

dazu, dass zwölf Zeilen aufgelistet werden. Die Angaben zu Plüderhausen, Schorndorf und Welzheim weisen Redundanzen auf, da die Attribute *ort* und *plz* von allen Fahrschülern in der Tabelle *fahrschueler* angezeigt werden.

schuelernr	nachname	vorname	ort
5	Emmrich	Anton	Adelberg
7	Fernandes	Enrico	Lorch
9	Grund	Dietmar	Plüderhausen
4	Deiß	Dagmar	Plüderhausen
11	Dressel	Andreas	Schorndorf
8	Lutz	Frederik	Schorndorf
1	Abele	Andreas	Schorndorf
6	Dressel	Dagmar	Schorndorf
3	Cramer	Conradt	Schorndorf
10	Demürel	Ali	Welzheim
2	Beutel	Barbara	Welzheim
12	Schmidt	Alex	Welzheim



Mit dem SQL-Schlüsselwort **DISTINCT** können doppelte Werte bei einem Abfrageergebnis verhindert werden. Es sorgt dafür, dass Datensätze, die bereits im Abfrageergebnis vorhanden sind, nicht wiederholt werden.

Dazu muss das Schlüsselwort DISTINCT direkt hinter dem SELECT platziert werden.

```
SELECT DISTINCT ort, plz
FROM fahrschueler;
```

➔ Abfrageergebnis:

ort	plz
Schorndorf	73614
Welzheim	73642
Plüderhausen	73655
Adelberg	73099
Lorch	73547

L1_5.7 Redundanzen in Abfrageergebnissen vermeiden – Lösungen

1
```
SELECT DISTINCT ort, plz
FROM fahrschueler;
```

2
```
SELECT DISTINCT vorname
FROM fahrschueler;
```

3
```
SELECT DISTINCT nachname
FROM fahrschueler;
```

4
```
SELECT DISTINCT fahrstundenzahl
FROM fahrschueler;
```

L1_5.8 Daten einer Datenbank mit SQL abfragen. Gruppierung – Aufgaben

Heiner Blechle möchte weitere Auswertungen seiner Datenbank vornehmen

Formulieren Sie für die folgenden Aufgabenstellungen die entsprechenden SQL-Befehle und dokumentieren Sie Ihre Lösungen.

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informationsmaterial *L1_5.8 Information Datenbankabfrage Gruppierung.docx*.

1 Es soll die Anzahl der Fahrschüler je Wohnort ermittelt und angezeigt werden.

2 Es soll ermittelt werden, wie viele Fahrstunden insgesamt je Wohnort der Fahrschüler genommen wurden.

3 Es soll die jeweilige Anzahl der Fahrschüler angezeigt werden, die aus Schorndorf bzw. aus Welzheim kommen.

4 Es soll die jeweilige Summe der Fahrstunden ermittelt werden, die von Fahrschülern aus Lorch bzw. Plüderhausen genommen wurden.

5 Es soll die Anzahl der Fahrschüler mit der gleichen Anzahl an Fahrstunden ermittelt werden, deren Fahrstundenanzahl weniger als vier Stunden beträgt.

6 Es soll die Anzahl der Fahrschüler je Wohnort, deren PLZ mit 736 beginnt, ermittelt werden.

7 Es soll die Anzahl der Fahrschüler je Wohnort ermittelt werden, aber nur von den Orten, aus denen mehr als zwei Fahrschüler kommen.

8 Es soll ermittelt werden, wie viele Fahrstunden insgesamt je Wohnort der Fahrschüler genommen wurden. Angezeigt werden sollen nur die Orte, deren Fahrschüler insgesamt mehr als 20 Fahrstunden genommen haben.

L1_5.8 Daten einer Datenbank mit SQL abfragen. Gruppierung – Information

GROUP BY - Klausel

Für die Auswertung von Datenbanken kann es oftmals hilfreich sein, die Ergebnisse einer Abfrage zu gruppieren, weil nicht nur einzelne Informationen, sondern auch Zusammenfassungen gewünscht werden. Durch die **GROUP BY**-Klausel im **SELECT**-Befehl werden alle Zeilen, die in einer oder mehreren Spalten den gleichen Wert enthalten, in jeweils einer Gruppe (Zeile) zusammengefasst.

Die GROUP BY-Klausel wird i.d.R. in Verbindung mit Aggregatfunktionen verwendet. Hinter dem Schlüsselwort SELECT dürfen nur dann Funktionen in Kombination mit weiteren Attributen verwendet werden, wenn von diesen Attributen mit der GROUP BY-Klausel Gruppen gebildet werden.

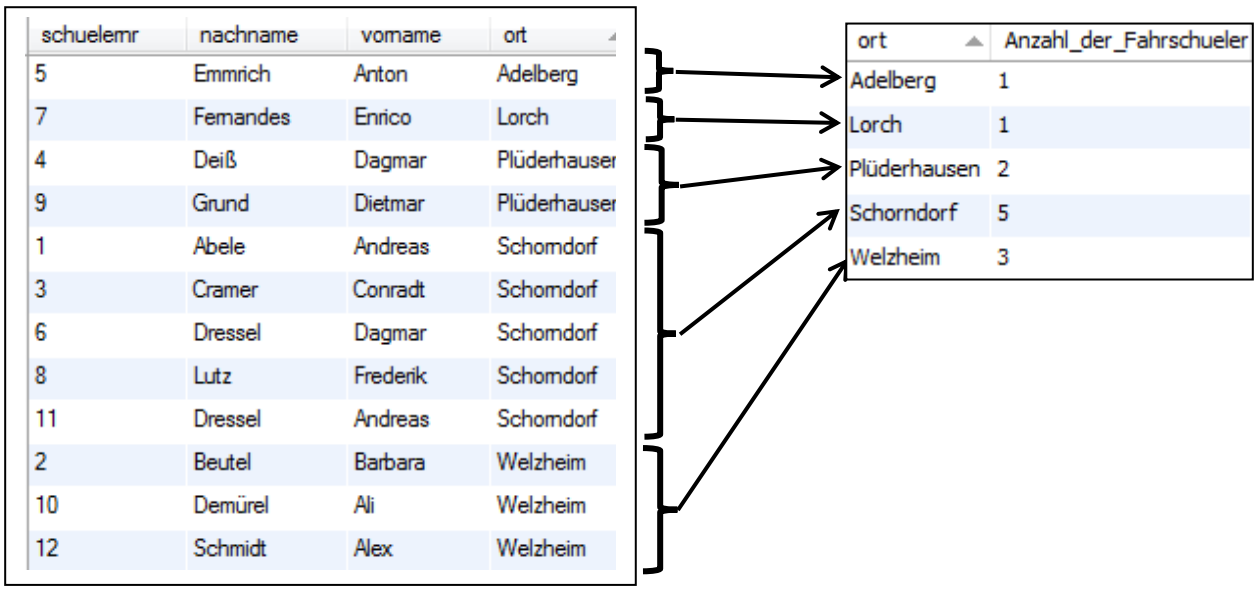
Soll beispielsweise ermittelt werden, wie viele Fahrschüler aus den jeweiligen Orten in der Datenbank erfasst sind, muss nach dem Ort und der Anzahl gezählter Datensätze projiziert werden.

Um ein aussagefähiges Ergebnis zu erhalten, muss das Attribut *ort* gruppiert werden. Es ist sinnvoll, für das Funktionsergebnis einen sogenannten Alias (= Andersname) zu vergeben. Dies erfolgt mit Hilfe des Schlüsselworts 'AS'.

Die SQL-Anweisung

```
SELECT ort, COUNT(*) AS Anzahl_der_Fahrschueler
FROM fahrschueler
GROUP BY ort;
```

führt dann zu folgendem Ergebnis:



schuelernr	nachname	vorname	ort
5	Emmrich	Anton	Adelberg
7	Fernandes	Enrico	Lorch
4	Deiß	Dagmar	Plüderhauser
9	Grund	Dietmar	Plüderhauser
1	Abele	Andreas	Schorndorf
3	Cramer	Conradt	Schorndorf
6	Dressel	Dagmar	Schorndorf
8	Lutz	Frederik	Schorndorf
11	Dressel	Andreas	Schorndorf
2	Beutel	Barbara	Welzheim
10	Demürel	Ali	Welzheim
12	Schmidt	Alex	Welzheim

ort	Anzahl_der_Fahrschueler
Adelberg	1
Lorch	1
Plüderhausen	2
Schorndorf	5
Welzheim	3

HAVING - Klausel

Die GROUP BY – Klausel kann mit der HAVING – Klausel erweitert werden. Sie ermöglicht es, eine gruppierte Ergebnismenge einzuschränken.

Mit der HAVING - Klausel wird festgelegt, welche der gruppierten Sätze angezeigt werden sollen. Mit ihr wird die Ausgabe einer Ergebnismenge auf Basis der Aggregatfunktionen AVG, COUNT, MAX, MIN und SUM eingeschränkt.
HAVING wirkt also nur zusammen mit GROUP BY und zeigt alle von GROUP BY gruppierten Datensätze an, welche die HAVING-Bedingung erfüllen.

Soll bspw. die Anzahl der Fahrschüler je Wohnort von den Orten ermittelt werden, aus denen mehr als zwei Fahrschüler kommen, muss das gruppierte Ergebnis noch nach der gewünschten Anzahl der Fahrschüler selektiert werden.
Hierzu wird die HAVING-Klausel in Verbindung mit GROUP BY verwendet

Die SQL-Anweisung

```
SELECT ort, COUNT(*) AS Anzahl_der_Fahrschueler
FROM fahrschueler
GROUP BY ort
HAVING COUNT(*) > 2;
```

führt dann zu folgendem Ergebnis:

ort	Anzahl_der_Fahrschueler
Schorndorf	5
Welzheim	3

Die Auflistung wird dabei automatisch nach dem Attribut aufsteigend sortiert, nach dem gruppiert wurde (hier: *ort*).

L1_5.8 Daten einer Datenbank abfragen. Gruppierung – Lösungen

1
```
SELECT ort, COUNT(*) AS Anzahl_der_Fahrschueler
FROM fahrschueler
GROUP BY ort;
```

2
```
SELECT ort, SUM(fahrstundenzahl) AS Fahrstunden_je_Ort
FROM fahrschueler
GROUP BY ort;
```

3
```
SELECT ort, COUNT(*) AS Anzahl_der_Fahrschueler
FROM fahrschueler
WHERE ort = 'Schorndorf'
OR ort = 'Welzheim'
GROUP BY ort;
```

4
```
SELECT ort, SUM(fahrstundenzahl) AS Fahrstunden_Lorch_Pluederhausen
FROM fahrschueler
WHERE ort ='Lorch'
OR ort ='Plüderhausen'
GROUP BY ort;
```

5
```
SELECT fahrstundenzahl, COUNT(*) AS Anzahl_der_Fahrschueler
FROM fahrschueler
WHERE fahrstundenzahl < 4
GROUP BY fahrstundenzahl;
```

6
```
SELECT plz, ort, COUNT(*) AS Anzahl_der_Fahrschueler
FROM fahrschueler
WHERE plz LIKE '736%'
GROUP BY ort;
```

7
```
SELECT ort, COUNT(*) AS Anzahl_der_Fahrschueler
FROM fahrschueler
GROUP BY ort
HAVING COUNT(*) >2;
```

8
```
SELECT ort, SUM(fahrstundenzahl) AS Fahrstunden_je_Ort
FROM fahrschueler
GROUP BY ort
HAVING SUM(fahrstundenzahl) > 20;
```

L1_6.1 Daten mit SQL verwalten - Daten einfügen – Aufgaben

1 In der Zwischenzeit hat sich die Fahrschule Blechle einen guten Ruf erworben, und es melden sich neue Fahrschüler an.

Folgende neue Fahrschülerin, die bereits **eine Fahrstunde** erhalten hat, soll in der Tabelle *fahrschueler* erfasst werden:

Schülernummer 13,
Susanne Cramer, wohnhaft in 73614 Schorndorf, Buchenweg 8,
Telefon: 01763 734572,
Email: susi.cramer@web.de,
geboren am 18.02.2002

Formulieren Sie eine SQL-Anweisung, mit der die genannten Daten in die Tabelle *fahrschueler* eingefügt werden.

Dokumentieren Sie Ihre Lösung.

Beachten Sie das Informationsmaterial *L1_6 Information Daten einfügen.docx*.

2 Fügen Sie in der Tabelle *fahrschueler* die Daten des folgenden Fahrschülers hinzu:

(Schülernummer 14)
Hakan Öztürk, wohnhaft in 73655 Plüderhausen, Sperbergasse 19, Telefon: 0155 2368992 (Rest unbekannt).

Formulieren Sie eine SQL-Anweisung, mit der die genannten Daten in die Tabelle *fahrschueler* eingefügt werden.

Dokumentieren Sie Ihre Lösung.

3 Fügen Sie in der Tabelle *fahrschueler* die Daten des folgenden Fahrschülers hinzu:

(Schülernummer 15)
Luigi Bellino, Goethestrasse 33, 73547 Lorch, 2 Fahrstunden - (Rest unbekannt).

Formulieren Sie eine SQL-Anweisung, mit der die genannten Daten in die Tabelle *fahrschueler* eingefügt werden.

Dokumentieren Sie Ihre Lösung.

L1_6 Daten mit SQL verwalten - Daten einfügen – Information

Die Datenbanksprache SQL enthält für das Einfügen von Daten in eine Datenbanktabelle den INSERT-Befehl.

Befehlssyntax der INSERT-Anweisung:

INSERT INTO *tabellenname*

[*attributname1*, *attributname2*, ….]

VALUES

(*wert1*, *wert2*, …..);

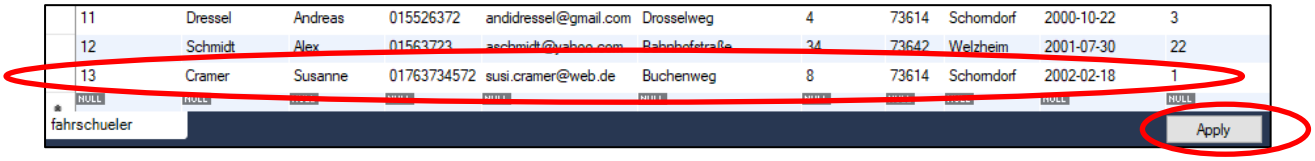
1. Zeile: Nach den SQL-Schlüsselwörtern INSERT INTO wird der Tabellenname der Tabelle genannt, in der die Daten eingefügt werden sollen.
2. Zeile: Hier können die Attributnamen (mit Komma getrennt) angefügt werden, denen Werte hinzugefügt werden sollen. Auf diese Angaben kann dann verzichtet werden, wenn alle Attribute Werte erhalten sollen. In diesem Fall müssen die konkreten Werte (Zeile 4) in der Reihenfolge der in der Tabelle festgelegten Attribute angegeben werden (hier: *schuelernr*, *nachname*, *vorname*, ….).
3. Zeile: Es folgt das SQL-Schlüsselwort VALUES.
4. Zeile: Die zu erfassenden Attributwerte werden in Klammern (mit Komma getrennt) nacheinander angegeben. Hierbei ist der Datentyp der jeweiligen Attribute zu beachten. Texte und Datumsangaben sind mit Hochkomma anzugeben.

Für die zu erfassenden Daten der neuen Fahrschülerin Susanne Cramer ergibt sich somit folgende Anweisung:

```
INSERT INTO fahrschueler
(schuelernr, nachname, vorname, telefon, email, strasse, hausnr, plz, ort, geburtsdatum, fahrstundenzahl)
VALUES
(13, 'Cramer', 'Susanne', '01763734572', 'susi.cramer@web.de', 'Buchenweg ', '8', '73614', 'Schorndorf', '1999-02-18', 1);
```

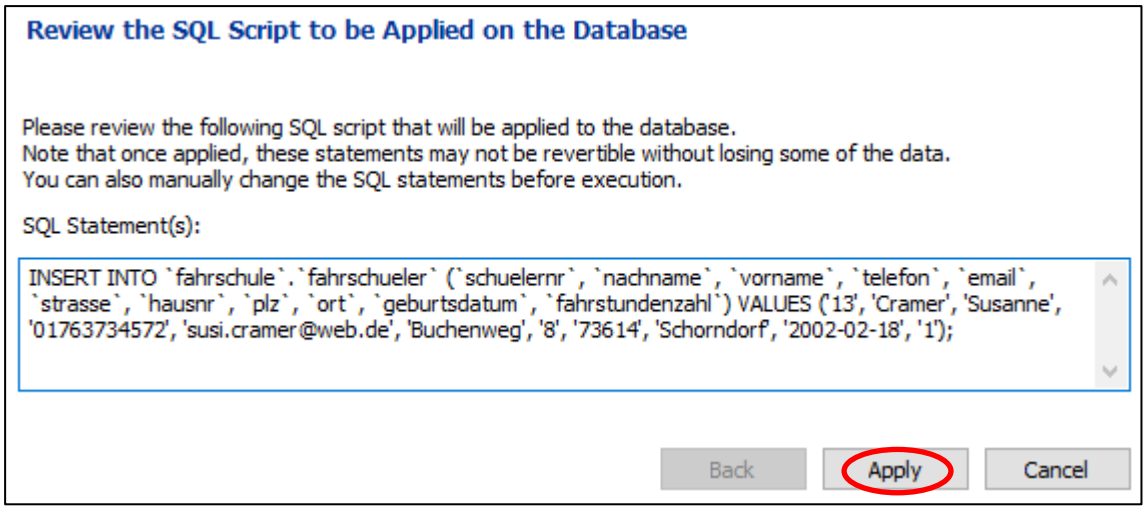
Alternative Vorgehensweise

Die MySQL-Workbench bietet die Möglichkeit, sich diesen Befehl generieren zu lassen. Dazu werden die Daten der neuen Fahrschülerin an die Liste angefügt, die aufgrund folgenden SQL-Befehles erscheint: *Select * from fahrschueler*;



Diese Eingabe muss mit der Schaltfläche *'Apply'* bestätigt werden.

Im nächsten Fenster wird der SQL-Befehl angezeigt, welcher aufgrund der Dateneingabe für die Fahrschülerin Susanne Cramer automatisch erzeugt wurde:



Nach nochmaliger Bestätigung der Schaltfläche *'Apply'* und danach der Schaltfläche *'Finish'* werden die Daten als zusätzlicher Datensatz in der Datenbanktabelle gespeichert.

L1_6.1 Daten mit SQL verwalten - Daten einfügen – Lösungen

1
```
INSERT INTO fahrschueler
(schuelernr, nachname, vorname, telefon, email, strasse, hausnr, plz, ort, geburtsdatum,
fahrstundenzahl)
VALUES
(13, 'Cramer', 'Susanne', '01763734572', 'susi.cramer@web.de', 'Buchenweg ', '8',
'73614', 'Schorndorf', '2002-02-18', 1);
```

2
```
INSERT INTO fahrschueler
(schuelernr, nachname, vorname, telefon, strasse, hausnr, plz, ort)
VALUES
(14, 'Öztürk', 'Hakan', '0552368992',  'Sperbergasse', '19', '73655', 'Plüderhausen');
```

3
```
INSERT INTO fahrschueler
(schuelernr, nachname, vorname, strasse, hausnr, plz, ort, fahrstundenzahl)
VALUES
(15, 'Bellino', 'Luigi',  'Goethestraße', '33', '73547', 'Lorch', 2);
```