Unternehmenswiki

„SQL“

Fach-Team:
Franz Kislik
Martin Pietsch
Tobias Stößl
Daniel Liebscher

Mit freundlicher Genehmigung des Bildungsservers Baden-Württemberg

Inhalt

1. Organisation in Dateien

Informationen zu sammeln und zu verwalten, stellt eine wesentliche Tätigkeit im Beruf, aber auch im Privatleben dar. Durch unkontrolliert wachsende Datenbestände ist jedoch in bestimmten Bereichen ein Datenchaos entstanden, das sich weiter ausbreitet und zu einem enormen Problem in Betrieben werden kann.

Ursache für dieses Datenchaos ist u. a. eine isolierte Datenhaltung für einzelne Anwendungen. So kann es in einem Betrieb vorkommen, dass Kundendaten für unterschiedliche Anwendungen (z. b. für Abrechnungen, für die Zusendung von Informationsmaterialien, für Service und Beratung) jeweils neu gespeichert werden. Durch Mehrfachspeicherung gleicher Informationen wird nicht nur Speicherplatz verschwendet, sondern auch die gesamte Informationsverarbeitung ineffizient.

Die schwerwiegendsten Fehler entstehen aber dadurch, dass bei Mehrfachspeicherung gleicher Daten in den einzelnen Anwendungen unterschiedliche Änderungen vorgenommen werden. Dadurch sind die Daten nicht mehr widerspruchsfrei.

Das Datenbanksystem bietet dabei dem Anwender die Möglichkeit, eine Struktur in die Datenbasis mittels einer Datendefinitionssprache zu bringen. Des Weiteren ist es einerseits für die Pflege von Daten, andererseits für die Gewinnung von Informationen (Abfrage von Daten) geeignet. Der Benutzer hat zugleich noch die Möglichkeit, den Zugang und den Zugriff auf Daten zu regeln sowie Daten zu übertragen.

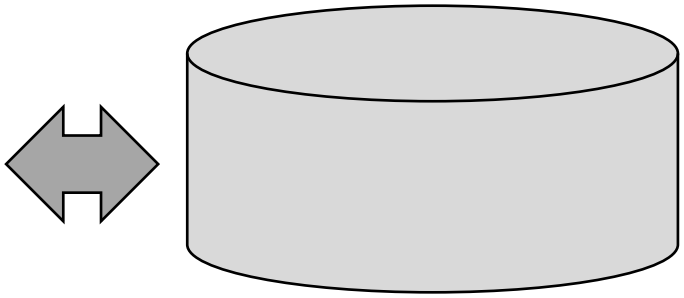
Während Excel die gesamte Datei, also den gesamten Datenbestand, bei jedem Öffnen in den Hauptspeicher lädt, erzeugt ein Datenbanksystem nur immer eine angestoßene Abfrage oder speichert einen Datensatz in die jeweiligen Tabellen. Mit anderen Worten lädt ein Datenbanksystem immer nur unmittelbar benötigte Daten aus der Datenbank in den Arbeitsspeicher. Dies ermöglicht erhebliche Performancevorteile gegenüber einer Excel Datenbank, da mit steigender Anzahl der Datensätze in Excel der verfügbaren Speicherkapazität Grenzen gesetzt werden. Des Weiteren kommt hinzu, dass mit zunehmenden Datenmengen die Zugriffszeit einer Excel Datenbank langsamer wird.

Ein Datenbanksystem ist eine Software, dies es erlaubt, große Datenmengen abzuspeichern, Daten nach beliebigen Kriterien zu durchsuchen, zu gruppieren und zu

verändern. Es besteht aus einem Datenbankmanagementsystem und einer Datenbasis und ist dabei wie folgt aufgebaut:

Datenbankmanagementsystem (DBMS):

- Definition (Aufbau der Datenstruktur)
- Manipulation (eingeben, ändern, löschen)
- Abfrage (Informationsgewinnung)
- Kontrolle (Zugangs- und Zugriffsrechte verwalten)



Datenbasis (Datenbank)

2. Begriffe einer Datenbank

Datenbank

Eine Datenbank ist der Rahmen, in dem zusammengehörige Entitätstypen in Tabellenform zusammengefasst sind. Alle Tabellen einer Firma, z.B. Kunden, LKWs, Rechnungen, Angestellte, Abteilungen, Lagerbestände usw. können in derselben Datenbank abgespeichert werden.

Nicht zusammengehörige Datenbestände werden in getrennten Datenbanken abgelegt, z.B. die Daten verschiedener Webshops eines ISPs.

Entität und Entitätsmenge

Jedes Objekt, das durch einen Datensatz abgebildet werden soll, nennt man Entität. Die Zusammenfassung gleicher Entitäten nennt man Entitätenmenge oder Entitätentyp. Z.B. unterscheidet sich der Entitätentyp Lastwagen vom Entitätentyp Kunde.

Hinweis: Die Entität bezeichnet das Objekt an sich, das je nach Umgebung verschieden beschreibbar sein kann. Zum Beispiel wird dieselbe Person (Entität) in der Verkehrssünderdatei in Flensburg mit anderen Merkmalen beschrieben als in einer Kundendatei.

Tabelle, Relation

Gleichartige Entitäten werden mit ihren Datensätzen in der gleichen Tabelle abgespeichert. Diese nennt man auch Relation. Daher ist auch der Name „Relationales Datenbanksystem“ abgeleitet. Jede Tabellenzeile bildet ein Objekt aus der realen Welt durch einen Datensatz ab.

Datensatz, Tupel

Die jeweils zusammengehörigen Daten, die eine Entität, z.B. eine Person, beschreiben, heißen Datensatz. Ein Datensatz, wird auch als Tupel bezeichnet, ein Begriff, der aus der Mathematik stammt. Analog zur Datensatzdefinition bezeichnet also ein Tupel (oder Datensatz) den Satz aller inhaltlich zusammengehörigen Datenfelder, z.B. Name, Vorname, Adresse einer Person.

Attribut

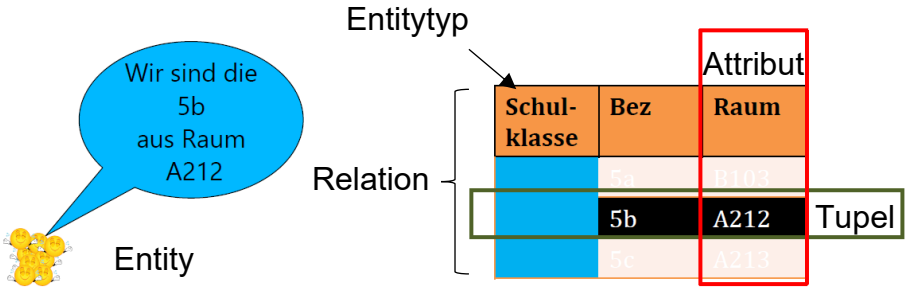
Eine Eigenschaft eines Entitätstyps nennt man Attribut, z.B. der Name eines Entitätentyps Kunde. Die Bezeichnung der Spalten (Attribute) wird auch als Spaltenüberschrift verwendet. Die Spalten der Tabellen entsprechen den Datenfeldern und die Zeilen, außer der Überschriftenzeile, entsprechen den Datensätzen.

Datenfeld

Die Attribute werden in Datenfeldern abgelegt. Man spricht z.B. vom Datenfeld Name in einem bestimmten Datensatz (s.u.), wenn das Attribut Name gemeint ist.

Datentyp

Attribute haben einen Typ, Datentyp (z.B. Zahl, Datum, Währung, Text oder ...) genannt, und eine Größe. Diese Vorgabe erleichtert das Abspeichern und die Fehlervermeidung.



3. Phasen der Datenbankentwicklung

3.1. Externe Phase

Es geht hierbei um die Ermittlung der Informationsstruktur: Die Datenbank soll einen Ausschnitt aus der realen Welt (auch als „Miniwelt“ bezeichnet) im Rechner abbilden. Diese Abbildung erfolgt durch die Beschreibung der Daten. Dazu ist der Informationsbedarf der Benutzer zu ermitteln und zu strukturieren. Das Ergebnis dieses ersten Schrittes – auch als Spezifikations- und Anforderungsanalyse bezeichnet – ist eine informelle Beschreibung des Fachproblems. Man überlegt sich, welche Daten für den benötigten Zweck notwendig sind. Der Datenbankentwickler muss dabei systematisch Vorgänge und Gegebenheiten der abzubildenden Realität untersuchen, damit wesentliche Aspekte nicht vergessen werden.

In dieser ersten Phase wird der Bedarf an Informationen gesammelt und analysiert. Dabei wird vor allem überprüft, welche realen oder abstrakten Objekte überhaupt in dieser „Miniwelt“ existieren. Die Ergebnisse werden vorwiegend in informalen Beschreibungen (Texte, tabellarische Aufstellungen, Formblätter usw.) des Fachproblems gesammelt.

Als Methoden zur Ermittlung des Informationsbedarfs werden Interviews, die Analyse bestehender Arbeitsprozesse sowie die Analyse existierender Dokumente (z.B. Formulare) eingesetzt. Dabei ist die externe Phase vom Datenbankmanagementsystem unabhängig. Diese oben genannten Objekte können im Beispiel einer Schule als „Miniwelt“ nicht nur reale Objekte wie Lehrer oder Schüler, sondern auch abstrakte Objekte wie beispielsweise Fächer, Zeugnisse oder Noten enthalten.

3.2. Konzeptionelle Phase

In dieser Phase wird ein sogenanntes semantisches Modell aufgestellt; dabei existieren verschiedene Ansätze zur Erzeugung einer solchen Gesamtsicht. Das bekannteste Modell ist das so genannte Entity-Relationship-Model (ER-Modell), welches eine grafische Beschreibung darstellt.

In dieser Phase werden die, in der externen Phase beobachteten Objekte (z. B. Lehrer, Schüler, Fächer) in einen Bedeutungszusammenhang gestellt. Das bereits erwähnte Entity-Relationship-Modell dient dazu, es entsprechend abzubilden.

Im konzeptionellen Entwurf wird die erste formale Beschreibung der Informationsstruktur des Anwendungsbereichs erstellt. Der konzeptionelle Entwurf vollzieht sich in den folgenden Teilschritten. Es wird also grafisch dargestellt, wie in der Schule die jeweiligen Gruppen untereinander agieren; so wird beispielsweise die Beziehung zwischen Schülern und Fächern, Schülern und Lehrern und auch Lehrer und Fächern dargestellt. Somit kann durch die grafische Darstellung abgelesen werden, wie die jeweiligen Gruppen miteinander verbunden sind. Auch hierbei ist es unerheblich, ob die jeweiligen Gruppierungen realer oder abstrakter Natur sind, die Beziehungen können zwischen allen Gruppierungsarten bestehen.

Dabei ist die konzeptionelle Phase vom Datenbankmanagementsystem unabhängig.

3.3. Logische Phase

Ziel ist die Übertragung des semantischen Datenmodells in ein logisches Datenmodell, z. B. in ein relationales Datenmodell (Tabellenform).

Diese Phase umfasst zwei Schritte:

1. Im ersten Schritt muss eine Transformation des konzeptionellen Schemas (ER-Modell) in das Datenbankschema erfolgen. Dazu wird das in der konzeptionellen Phase gebildete ER-Modell aufgelöst und in Tabellen überführt. Eine Tabelle wird dabei aus den jeweiligen Gruppierungen gebildet wie z.B. eine Tabelle „Lehrer“ oder „Schüler“.

2. Im zweiten Schritt erfolgt eine Optimierung des relationalen Schemas, z. B. die Eliminierung redundanter Daten (Normalisierung). Beispielsweise ist zu einem Schüler der Wohnort Fürth hinterlegt. Somit muss es bei jedem Schüler einzeln (!) eingegeben werden, was einen enormen Aufwand aufstellt; man denke hierbei auch an zukünftige Änderungen des Wohnorts. Unter Umständen werden bestimmte Schüler auch gar nicht in der Abfrage angezeigt, da es beispielsweise durch Rechtschreibfehler zu einer Falscheingabe gekommen ist (Bsp.: „Führt“).

Einer der wichtigsten Aufgaben der logischen Phase ist, die Art der Beziehung zwischen den jeweiligen Gruppen festzulegen, also den sogenannten Beziehungstyp zu bestimmen.

Dabei ist die logische Phase vom Datenbankmanagementsystem unabhängig.

3.4. Physische Phase

Die Ausgangslage ist folgende:

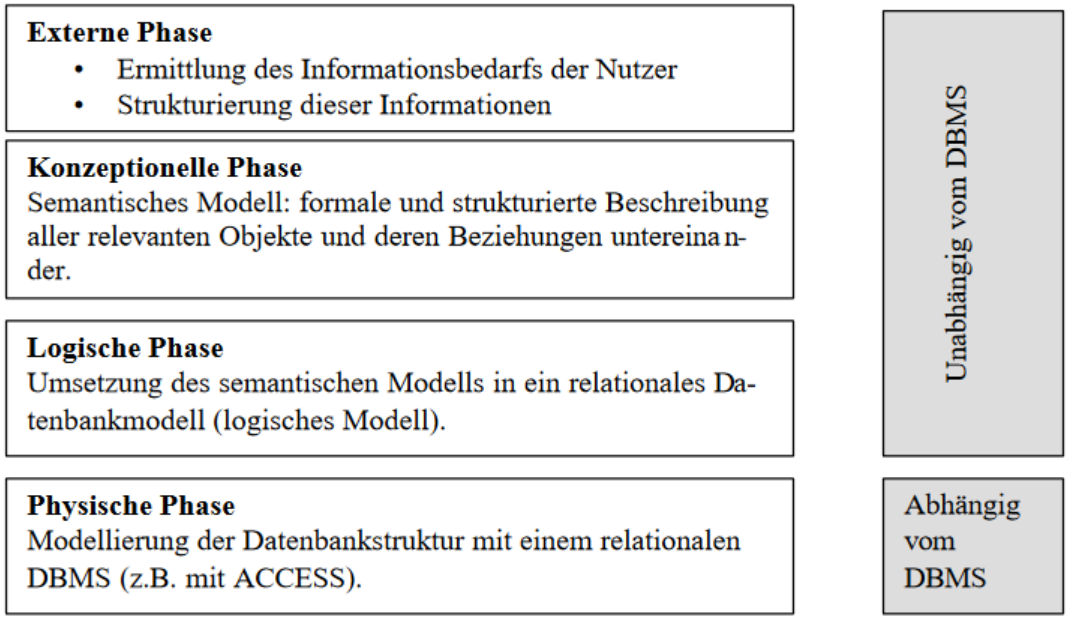
Die externe, konzeptionelle und die logische Phase sind bereits erstellt, der nächste Schritt ist der physische Entwurf der Datenbank, wobei am Ende dieser Phase die Datenbank existiert. Dazu wird das logische Modell in ein konkretes Datenbankschema übersetzt werden.

Dazu müssen Datentypen, Wertebereiche, Relationen und Gültigkeitsregeln festgelegt werden.

Eine dieser Gültigkeitsregeln könnte beispielsweise sein, dass bereits bei der Eingabe neuer Schülerdaten nur „Herr“ und „Frau“ ausgewählt werden kann; die Bezeichnung „Fräulein“ wird somit von vornherein ausgeschlossen. Um dabei beispielsweise die Eingabe für Mitarbeiter in der Schulverwaltung zu vereinfachen, können in dieser Phase Formulare erstellt werden, die die Eingabe von Schülerdaten vereinfachen.

In dieser Phase erfolgt der Entwurf des physischen oder internen Schemas. Für das definierte logische Schema werden nun geeignete Speicherungsstrukturen angelegt.

Es erfolgt eine Modellierung der Datenbankstruktur mit einem relationalen Datenbankmanagementsystem wie beispielsweise Access. Dadurch kann die Effizienz im gesamten Arbeitsprozess gesteigert werden, da nun bestimmte Informationen gefiltert und ausgewertet werden können. Die physische Phase ist dabei vom Datenbankmanagementsystem abhängig.



4. SQL

Die Abkürzung SQL steht für **S**tructured **Q**uery **L**anguage. SQL ist eine standardisierte Datenmanipulations- und Datenabfragesprache, die alle erforderlichen Sprachelemente enthält, um sämtliche Arbeiten, die beim Umgang mit einer relationalen Datenbank anfallen, auszuführen. Sie stellt damit die Schnittstelle vom Benutzer oder vom Anwendungsprogramm zur Datenbank dar.

SQL ermöglicht es …

- Datenbanken anzulegen,
- Tabellen einzurichten, zu verändern und zu löschen,
- Datensätze zu suchen, zu manipulieren und zu löschen,
- komplette Datenbanken zu sichern und zu verwalten,
- Datenbankenbenutzer und deren Zugriffsrechte auf Datenbanken, Tabellen oder Tabellenspalten einzurichten und zu verwalten.

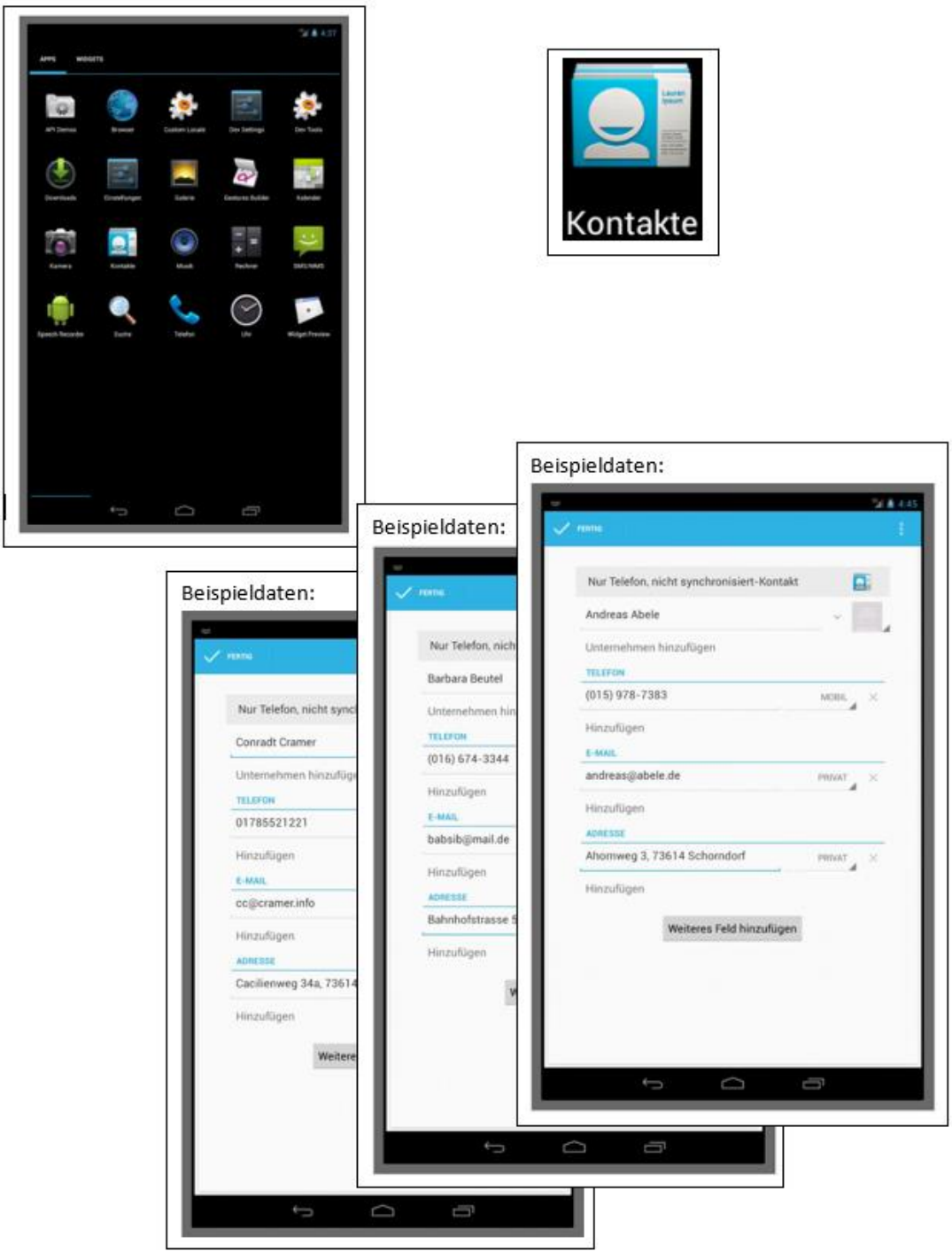
SQL ist aufgeteilt in **vier Befehlsgruppen**:

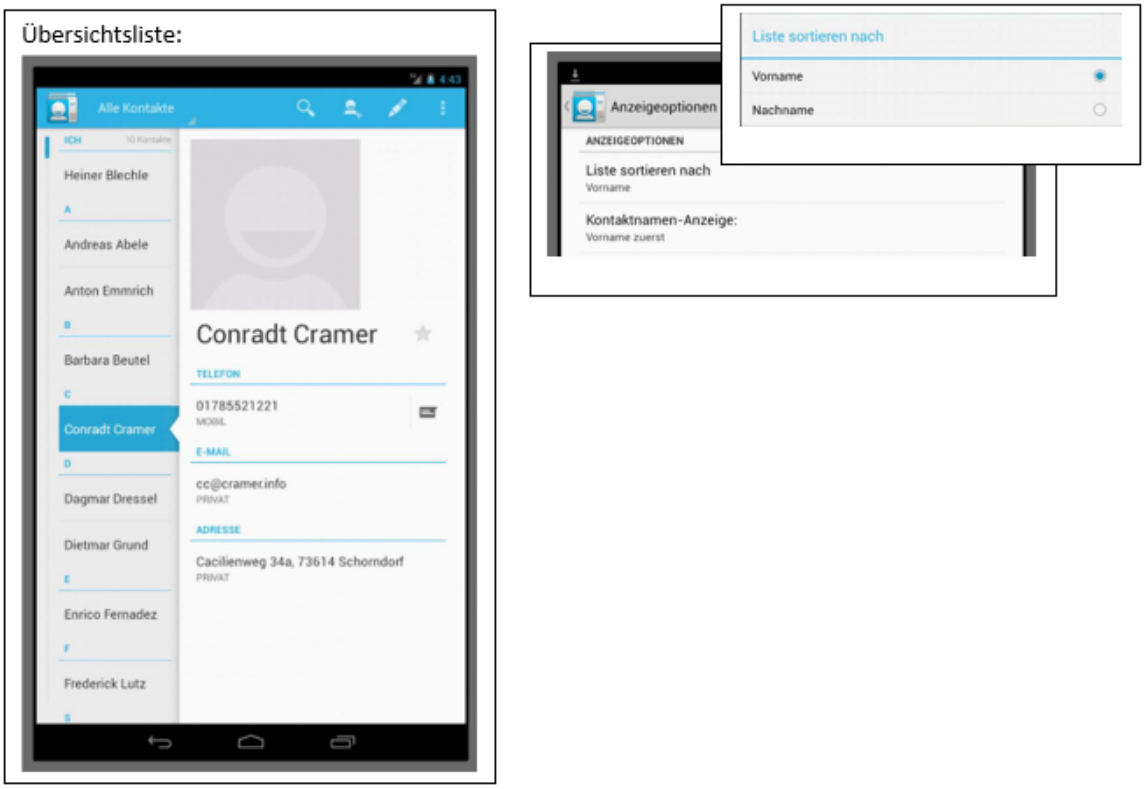
Data **D**efinition **L**anguage	DDL
Data **M**anipulation **L**anguage	DML
Data **Q**uery **L**anguage	DQL
Data **C**ontrol **L**anguage	DCL

5. Lernfortschritt 1

5.1.1.1 Ein Datenbankmodell mit einer Tabelle erstellen

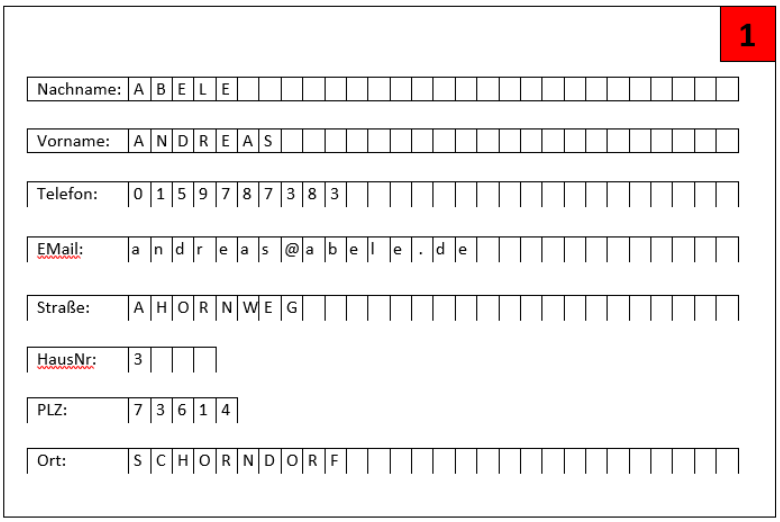
Heiner Blechle hat sich seinen beruflichen Traum erfüllt: Er hat seine Fahrlehrerprüfung bestanden und macht sich nun mit einem gebrauchten Fahrschulauto selbständig. Über seinen Sportverein hat er zu Jugendlichen und jungen Erwachsenen bereits Kontakte geknüpft, die bei ihm die Fahrschule besuchen möchten.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

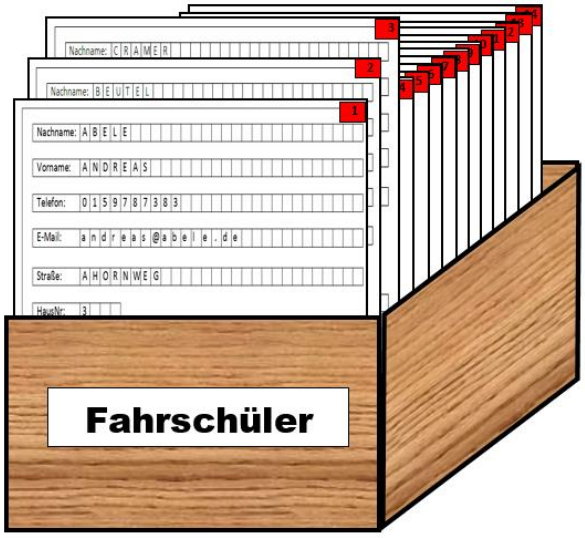




Heiner Blechle stellt bald fest, dass die Speicherung der Daten seiner Kunden allein auf dem Handy unpraktisch ist. So braucht er diese Informationen regelmäßig für Meldungen an die Führerscheinstelle bei der Kreisverwaltung, den TÜV für Fahrprüfungen aber auch für die Erstellung von Rechnungen. Außerdem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

Alexander, ein Freund 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.





Alexander empfiehlt dafür eine Datenbank zu entwickeln. Dazu muss als erster Schritt eine Tabelle für die gespeicherten Kontaktdaten der Fahrschüler entwickelt werden.

- Die Tabelle Fahrschüler bezeichnet man als Entitätstyp Fahrschüler.
- Jeder einzelne Fahrschüler entspricht einer Zeile (Fachbegriff Entität, Datensatz oder Tupel).
- Jede Zeile besteht aus mehreren Spalten (Fachbegriff Attribute).
- Jedes Attribut enthält die Einzelinformationen eines Fahrschülers (z.B. Nachname, Vorname, etc.).
- Für jedes Attribut muss festgelegt werden, ob darin Texte (Buchstaben, Ziffern oder Sonderzeichen) oder Zahlenwerte, wenn damit gerechnet wird, gespeichert werden sollen, d.h. es muss der jeweilige Datentyp festgelegt werden.
- Für jedes Attribut vom Datentyp Text muss die maximale Zeichenanzahl festgelegt werden.

5.1.1.2 Ein Datenbankmodell mit einer Tabelle erstellen

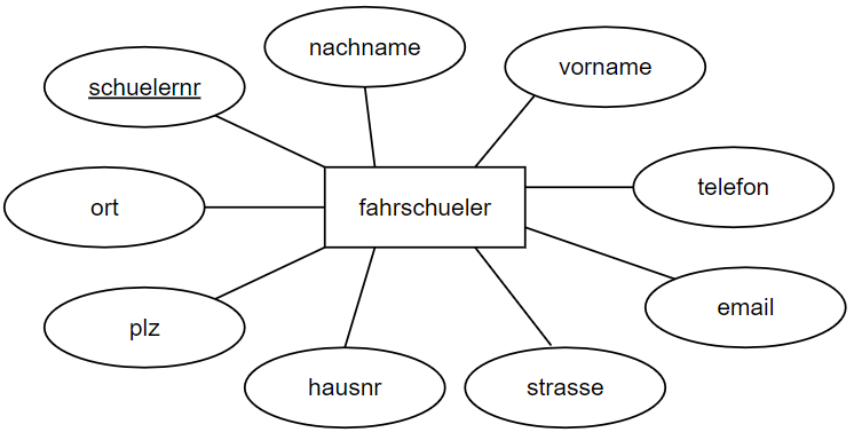
Heiner Blechle zeigt Alexander stolz seinen Tabellenentwurf. Alexander erklärt, dass jede Tabelle ein Attribut enthalten muss, das jeden Datensatz und somit jeden Fahrschüler eindeutig identifiziert (= Primärschlüssel). Es muss sichergestellt werden, dass der Wert dieses Attributs in allen Datensätzen der Tabelle nur ein einziges Mal vorkommt (wie z.B. eine Personalausweisnummer). Als Datentyp eines Primärschlüssels sollte eine ganze Zahl (= Integer, abgekürzt INT) verwendet werden. Bei INT braucht man keine Angabe zur maximalen Zeichenanzahl zu machen.

- den Namen des Primärschlüssels: schuelernr
- den Datentyp des Primärschlüssels: INT

Dadurch ergibt sich folgender Tabellenentwurf:

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

Und das passende ER-Diagramm:



Künftig verwenden wir folgende Datentypen:

Typ	Bedeutung	Speicherplatz	Beispiel	Besondere Schreibweise
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	
DATE	Datum von 01.01.1000 bis 31.12.9999	3 Bytes	Geburtsdatum: 1991-11-30	YYYY-MM-DD
TIME	Zeit	3 Bytes	Zielankunft: 03:56:04	hh:mm:ss

5.1.2 Ein ER-Modell (ER-Diagramm) erstellen

Alexander erklärt, dass man in der Softwareentwicklung zunächst ein Datenbank-Modell (=Design) der zu entwickelnden Softwarelösung erstellt.

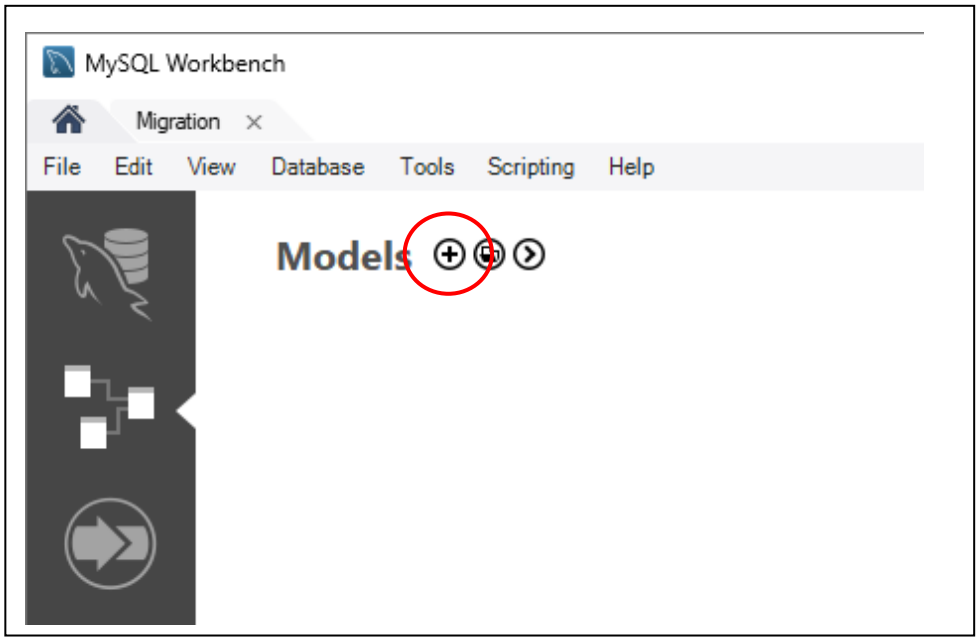
Im Bereich der Datenbanken verwendet man für die Modellierung ein Entity-Relationship-Modell (ER-Modell). Es wird auch Entity-Relationship-Diagramm (ER-Diagramm) genannt. Im Fall der Fahrschule besteht das ER- Diagramm lediglich aus dem bisher erstellten Tabellenentwurf.

Alexander schlägt vor, für die softwaregestützte Erstellung eines ER-Diagramms das Programm **MySQL-Workbench** zu verwenden.

Dafür sind die folgenden Schritte erforderlich:

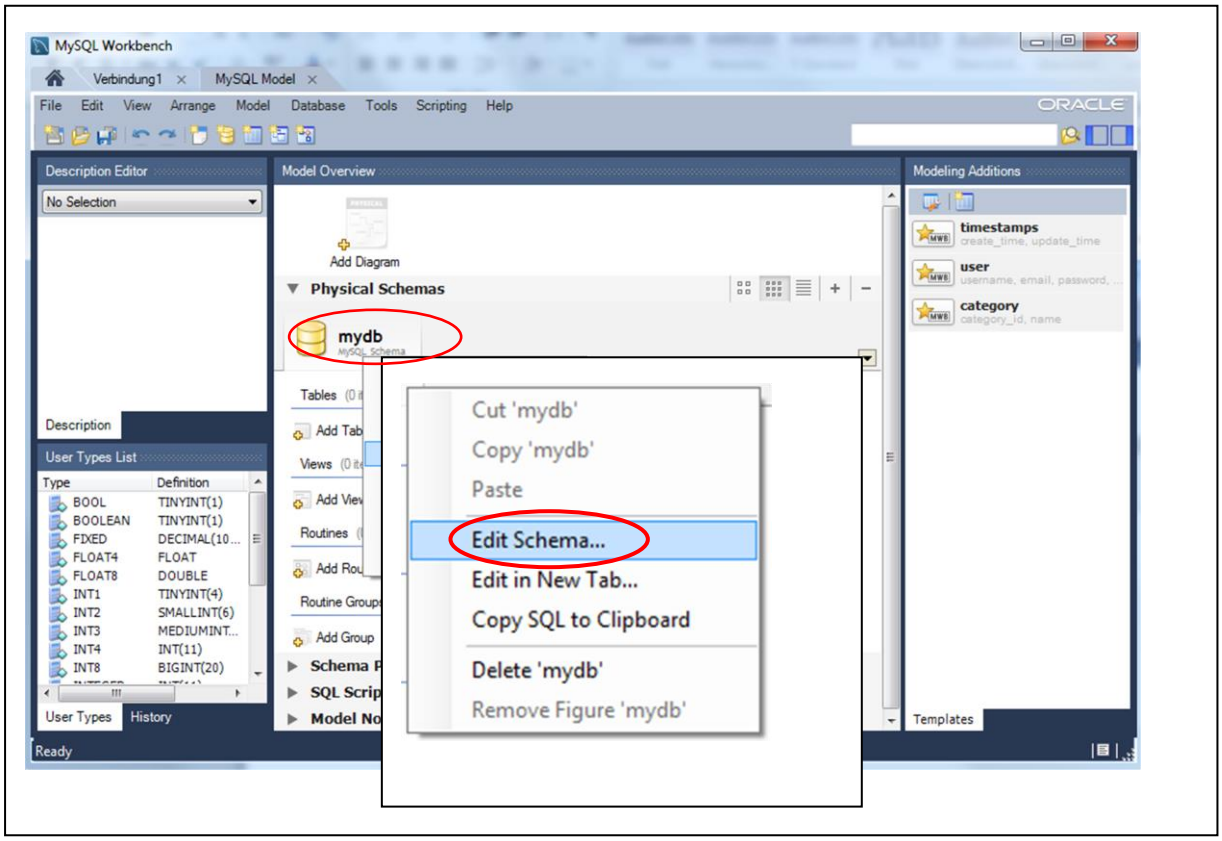
a) Starten der MySQL-Workbench

b) Erstellen eines neuen Modells



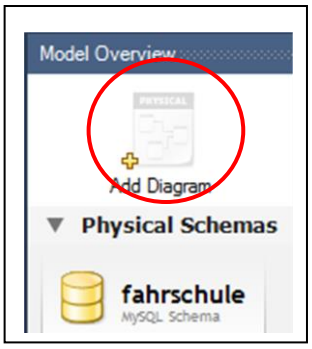
- Klick mit rechter Maustaste auf die bereits als Standard existierende Datenbank (= Schema) **mydb**

- **Edit Schema**

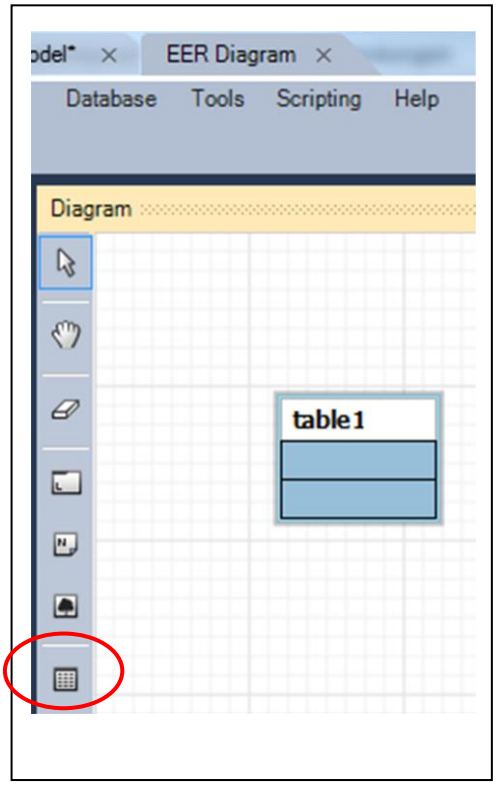


- Den Namen von **mydb** zu **fahrschule** ändern (Kleinschreibung beachten!)

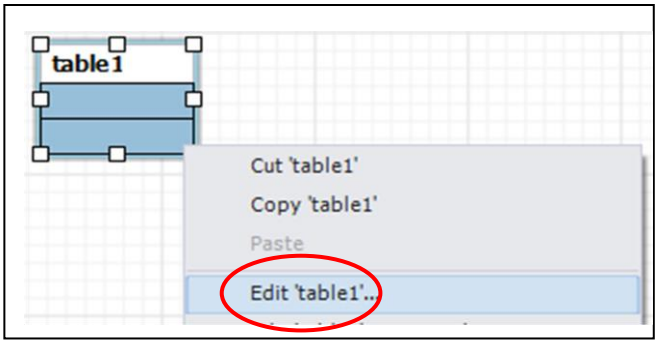
- Per Doppelklick auf **Add Diagram** ein ER-Diagramm erstellen. (Hinweis: Die MySQL-Workbench verwendet die Bezeichnung EER-Diagramm.)



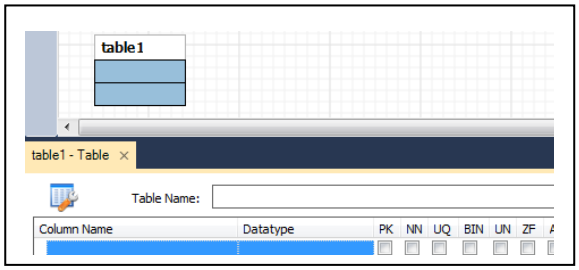
- Tabelle als Symbol wählen und auf die Zeichnungsfläche klicken.



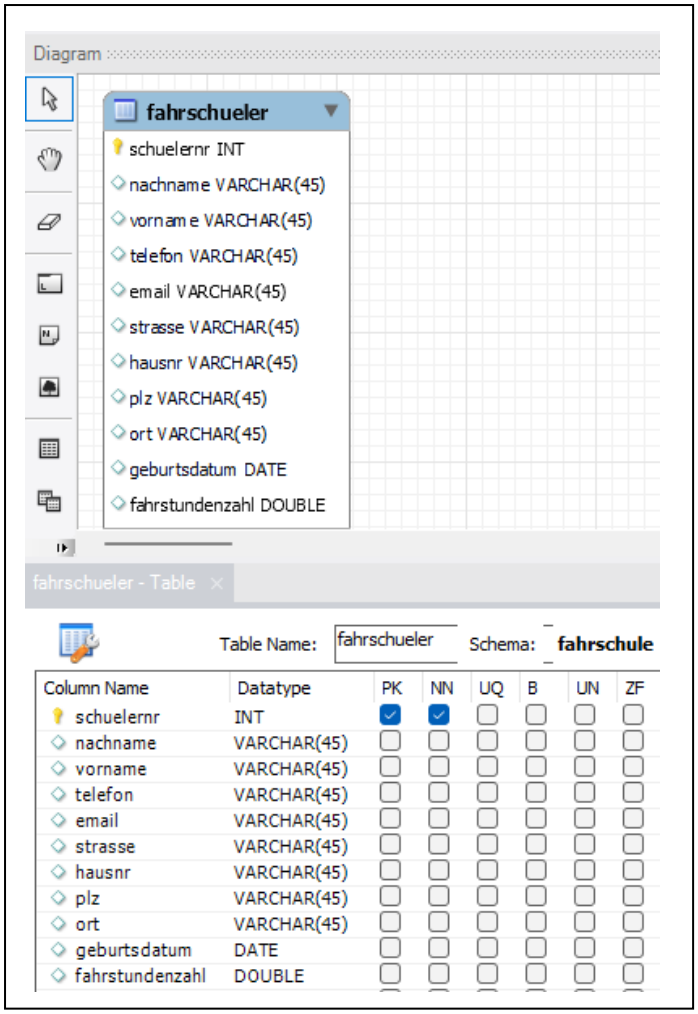
- Klick mit rechter Maustaste auf die gezeichnete Tabelle und Auswahl des Befehls **Edit table1**



- Editieren Sie den Tabellenentwurf:



Endergebnis



PK bedeutet **PrimaryKey** (= **Primärschlüssel**)

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 **Standard** ist **45**.

Heiner Blechle fällt auf, dass er bisher noch keine Speicherung des Geburtsdatums möglich ist. Zusätzlich wünscht er sich ein Attribut, um die Anzahl der Fahrstunden zu speichern.

- **Ergänzen** Sie die Tabellenstruktur entsprechend:

Column Name	Datatype	PK	NN
schuelernr	INT(11)	☑	☑
nachname	VARCHAR(45)	☐	☐
vorname	VARCHAR(45)	☐	☐
telefon	VARCHAR(45)	☐	☐
email	VARCHAR(45)	☐	☐
strasse	VARCHAR(45)	☐	☐
hausnr	VARCHAR(4)	☐	☐
plz	VARCHAR(5)	☐	☐
ort	VARCHAR(45)	☐	☐
geburtsdatum	DATE	☐	☐
fahrstundenzahl	DOUBLE	☐	☐

- **Speichern** Sie anschließend per Klick auf das **Diskettensymbol** das ER-Diagramm unter dem Namen **fahrschule_1_1.mwb** auf ihrem USB-Stick.

- **Schließen** Sie daraufhin die **Workbench**!

5.1.3 Aus einem ER-Diagramm eine Datenbank generieren

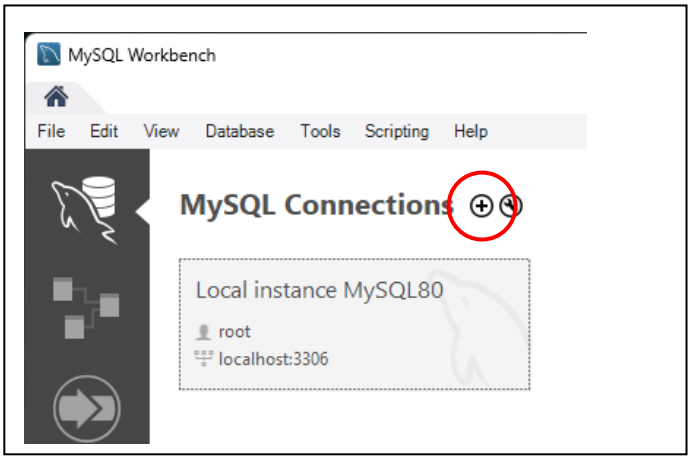
Auf der Grundlage der Modellierung soll nun eine Datenbank generiert werden, um Daten speichern zu können. Deshalb muss auf einem Datenbank-Server eine neue Datenbank entsprechend der Struktur des ER-Diagramm generiert werden.

Hierfür sind die folgenden Schritte erforderlich:

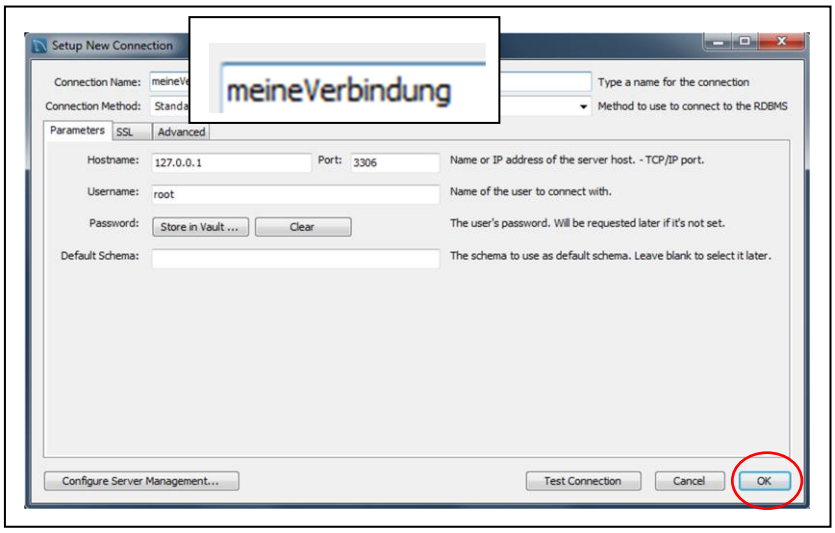
a) Starten des **MySQL-Servers**

b) Starten der **Workbench**

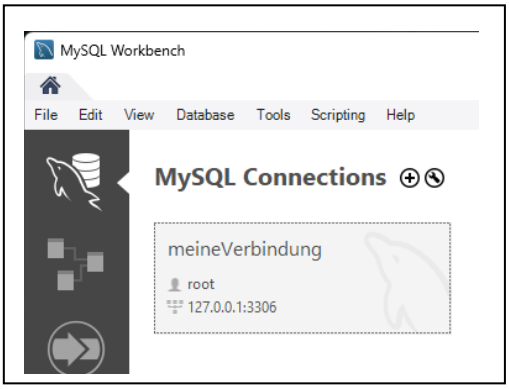
c) Erstellen einer **Verbindung** von der Workbench zum MySQL-Server.



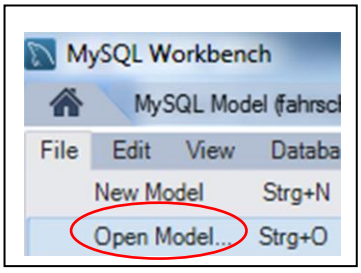
d) Eingabe eines **Verbindungsnamens**



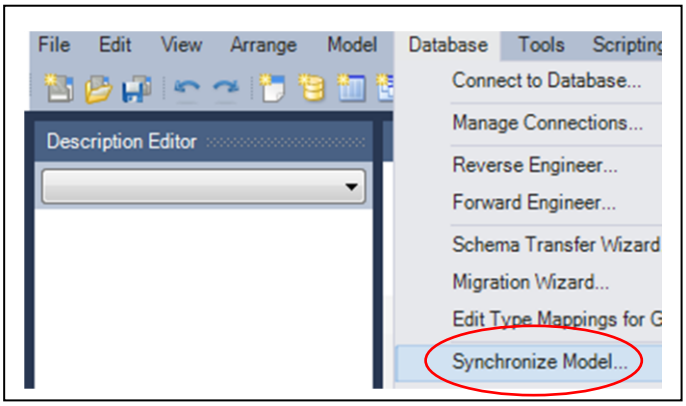
Über diese **erstellte Verbindung** kann man später auf die Datenbank zugreifen.



e) Öffnen Sie über das **Menü File / Open Model** das vorher gespeicherte Modell **fahrschule_1_1.mwb**.

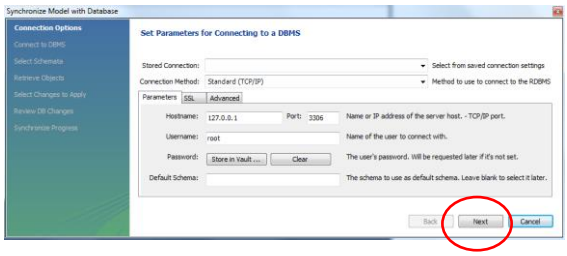


f) Mit dem **Menüpunkt Database / Synchronize Model** wird das **erstellte Modell in** eine **Datenbank** auf dem MySQL-Server **umgewandelt**.

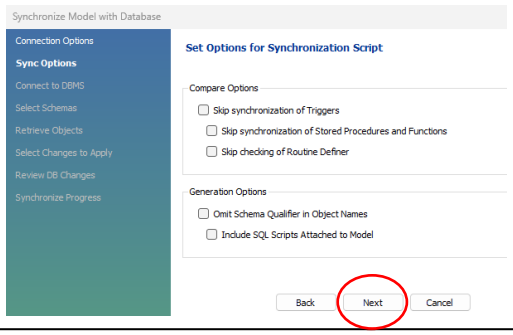


g) Die **nachfolgenden Schritte** werden jeweils **ohne weitere Eingaben** mit einem **Klick auf Next** bestätigt und der Vorgang im **letzten Schritt** mit **Close** abgeschlossen.

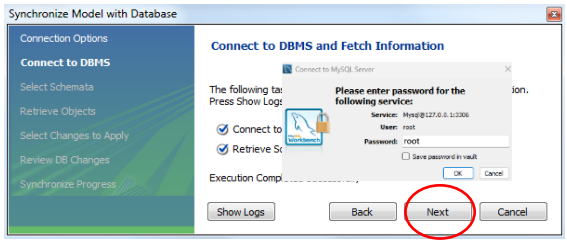
1.



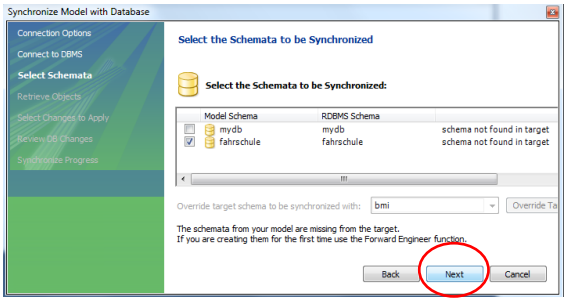
2.



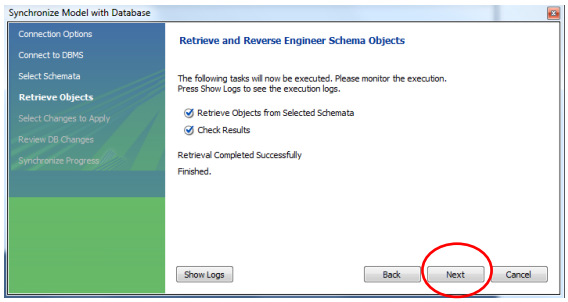
3.



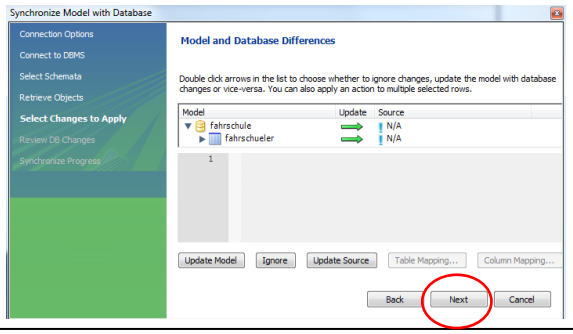
4.



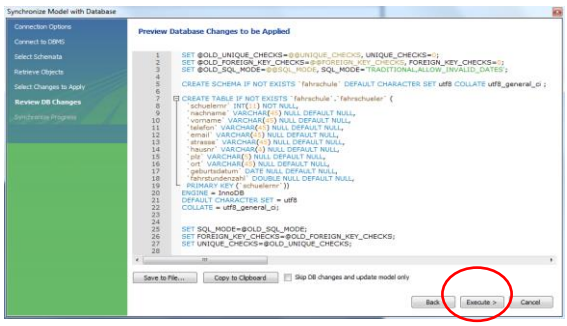
5.



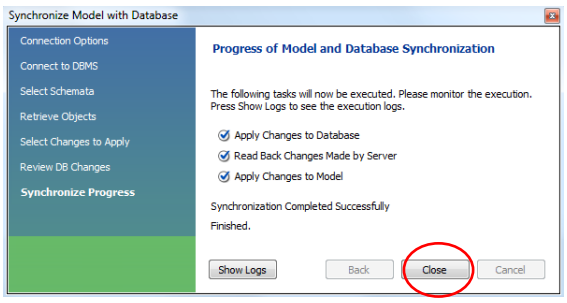
6.



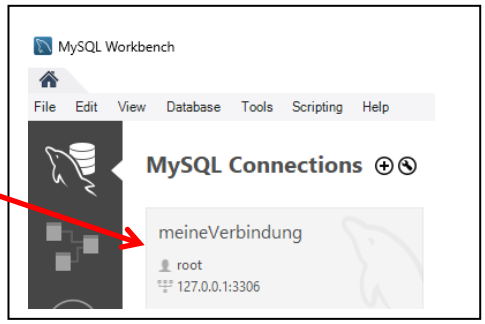
7.



8.



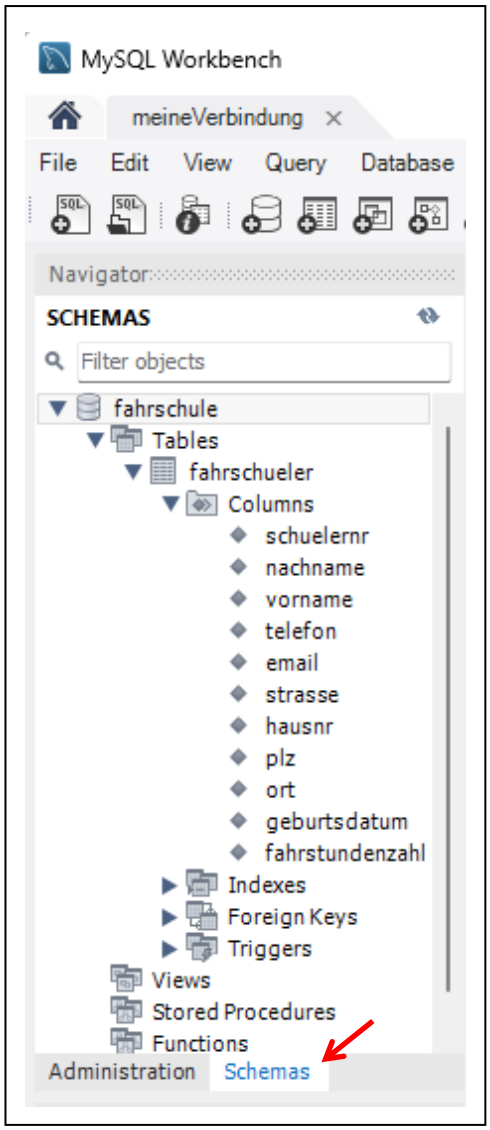
Öffnen der bestehenden Datenbankverbindung **mit** einem **Klick auf meineVerbindung**



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.



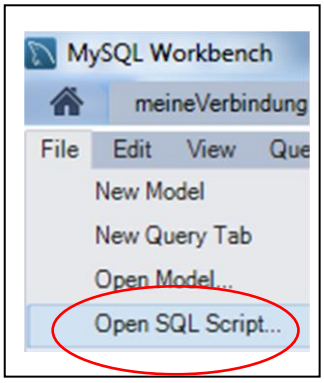
Wir werfen einen Blick hinter die Kulissen, um zu erkennen, wie die Workbench diese MySQL-Datenbank erzeugt hat:

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 fahrschu-
12-	Namen und Datentyp der Attribute. Nicht-Schlüssel-Attribute können nichts
23	Der Primärschlüssel ist das Attribut schuelernr.

5.1.4.1 Daten per SQL-Script einfügen und mit SQL anzeigen lassen



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

Vorgehensweise:

a) Das Script **L1_4_fahrschule_1_Tabelle_Daten_einfügen.sql** öffnen

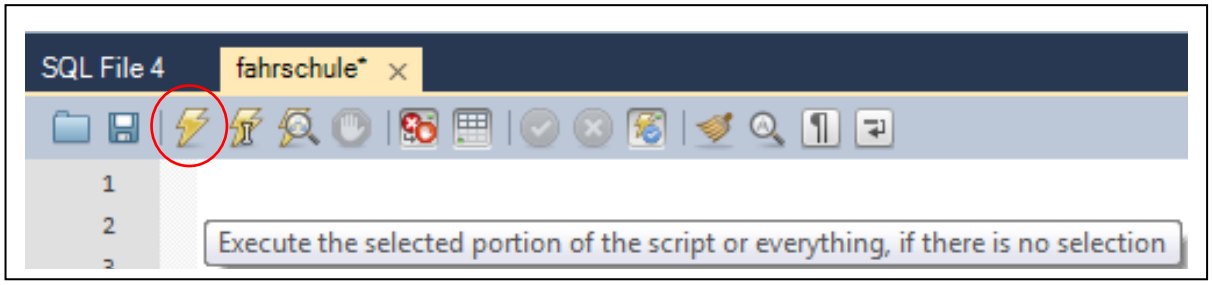
Das Script hat u.a. folgenden Inhalt:

```
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','Drosselweg','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:

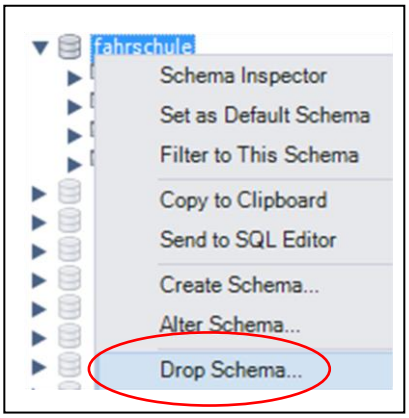
INSERT INTO `fahrschueler` VALUES (1,'Abele','Andreas','0159787383','andreas@abele.de','Ahornweg','3','73614','Schorndorf', '1999-12-02',2),
Füge einen Datensatz ein in der Tabelle fahrschueler mit den Werten (1,'Abele','Andreas','0159787383','andreas@abele.de','Ahornweg','3','73614','Schorndorf', '1999-12-02',2),

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

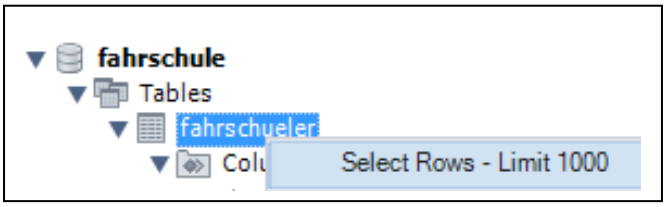


Hinweis:

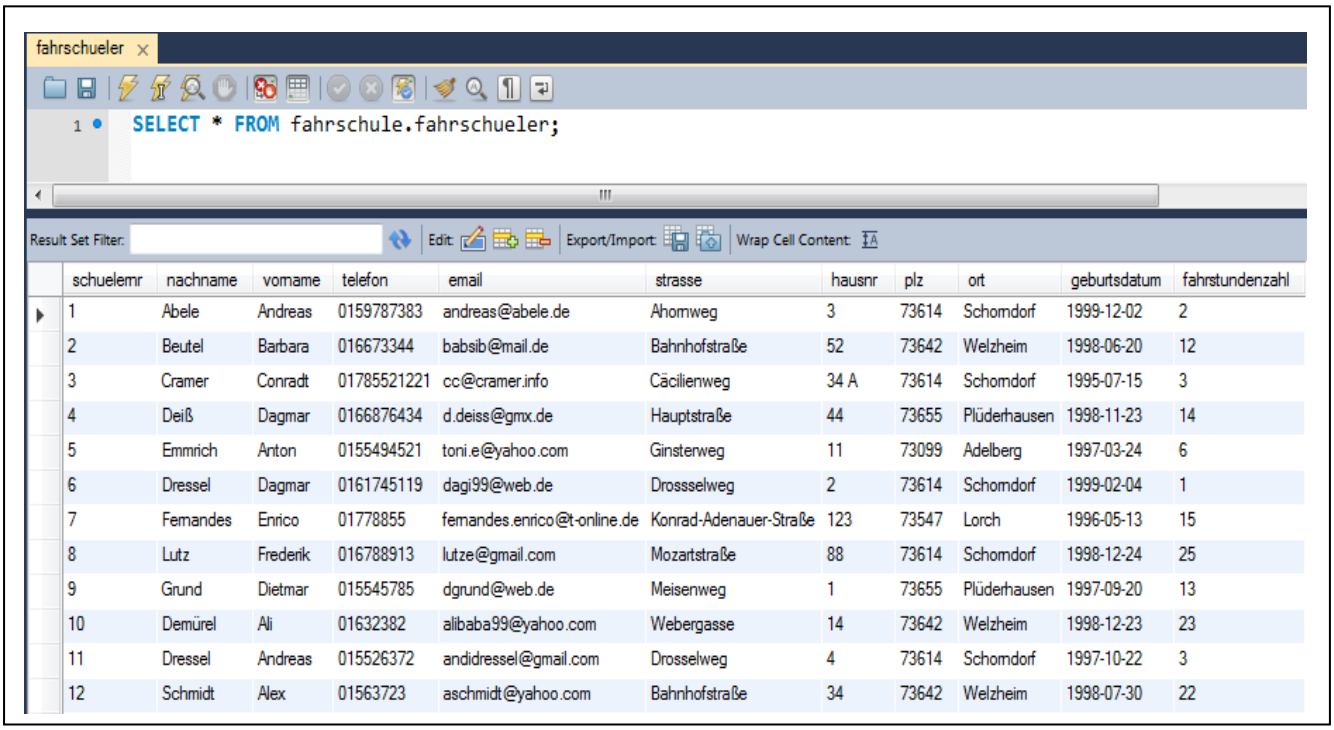
Für den Fall, dass bei Aufgabe **b)** Fehler auftreten kann mit Hilfe des Scripts **L1_4_fahrschule_1_Tabelle_komplett.sql** die vollständige Datenbank mit allen erforderlichen Tabellen und Daten erzeugt werden. Zuvor muss die bisherige Datenbank **fahrschule** gelöscht werden. (Dazu bitte die Lehrkraft kontaktieren.)



c) Die **gesamte Tabelle** mit den Datensätzen **anzeigen**
(Rechtes Maustastenmenü)



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	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	1996-05-13	15
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

SELECT * FROM fahrschule.fahrschueler;
Wähle alle Attribute 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 fettgeschrieben erscheint.

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

```
SELECT *
FROM fahrschueler;
```

Ein SQL-Befehl wird grundsätzlich mit einem Semikolon abgeschlossen!

5.1.4.2 Daten mit SQL abfragen - Befehlssyntax der Auswahlanweisung SELECT

SELECT <attributsbezeichnung1>, <attributsbezeichnung2> … **Projektion** (*= Alle Felder

FROM <tabelle> **Herkunftstabelle**

[**ORDER BY** <attributsbezeichnung1>,

<attributsbezeichnung2> … [ASC|DESC]..] **Sortieren (ASC bzw. DESC)**

Beispiel:

- **SELECT** nachname, vorname, email, ort
- **FROM** fahrschueler
- **WHERE** ort = “Welzheim”
- **ORDER BY** nachname DESC **;** ← Wichtig!

fahrschueler	nachname	vorname	email	ort
	Schmidt	Alex	aschmidt@yahoo.com	Welzheim
	Demürel	Ali	alibaba99@yahoo.com	Welzheim
	Beutel	Barbara	babsib@mail.de	Welzheim

Übersicht zu SQL - Klauseln

Die **FROM -** Klausel	Hinter **FROM** steht der Name der Tabelle
Die **ORDER BY** - Klausel	Daten werden nach einem oder mehreren Feldnamen **sortiert** ausgegeben. Die **vorgegebene Sortierreihenfolge ist aufsteigend ASC** (muss nicht angeben werden). Soll absteigend sortiert werden, muss **DESC** eingegeben werden.

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

1. Alle Fahrschüler mit Fahrschülernummer, Vor- und Nachnamen.

```
SELECT schuelernr, vorname, nachname
FROM fahrschueler;
```

Hinweis: Sollen nur bestimmte Attributwerte dargestellt werden, so müssen diese einzeln hinter SELECT aufgeführt werden.

2. Alle Fahrschüler mit Nachnamen, Email-Adresse und Wohnort.

```
SELECT nachname, email, ort
FROM fahrschueler;
```

B) Beispiele für SQL-Befehle zur Ausgabe nach einer gewünschten Reihenfolge:

1. Alle Fahrschüler mit Fahrschülernummer, Vor- und Nachnamen nach Nachname geordnet (sortiert).

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

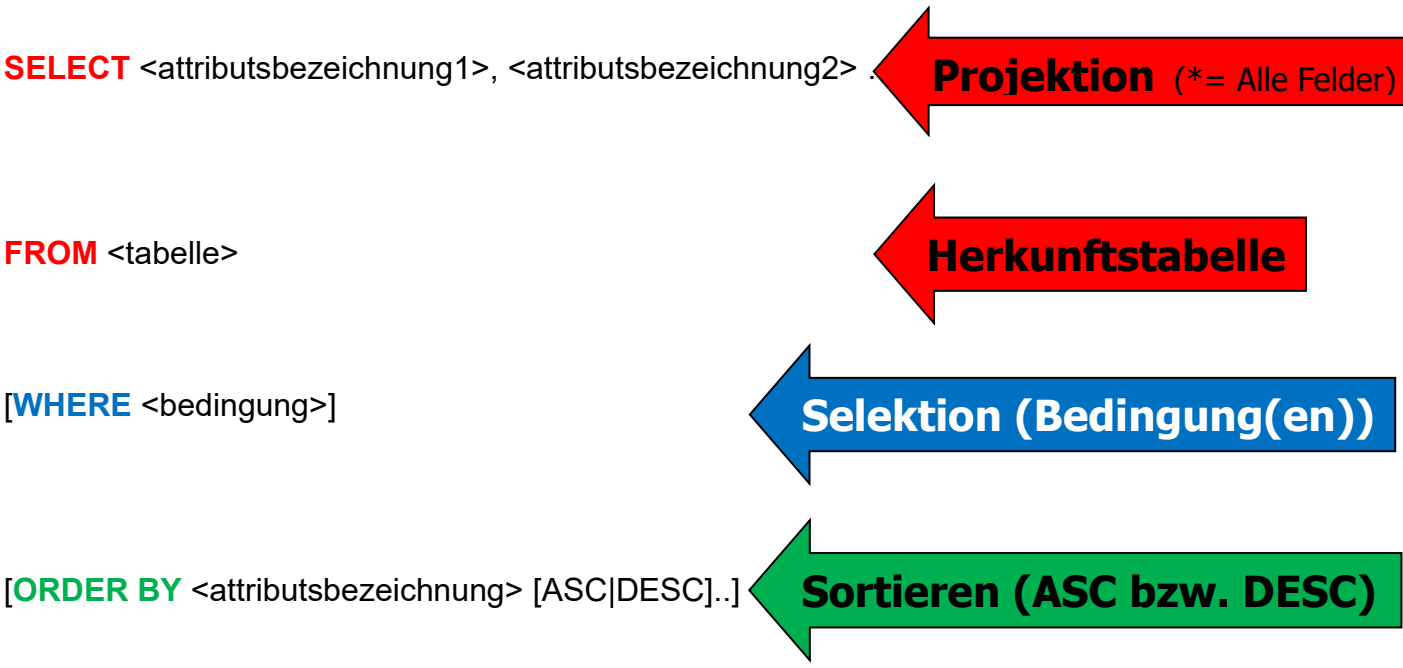
2. Alle Fahrschüler mit Fahrschülernummer, Vor- und Nachnamen und Wohnort nach Wohnort absteigend geordnet (sortiert).

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

3. Alle Fahrschüler mit Fahrschülernummer, Vor- und Nachnamen nach Nachname und anschließend nach Vornamen geordnet (sortiert).

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

5.1.4.3 Daten mit SQL abfragen - Befehlssyntax der Auswahlanweisung SELECT



Übersicht zu 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 logischen Operatoren AND, OR und NOT verknüpft.

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 linksstehende Vergleichswert kleiner ist
>	größer als	wahr, wenn der linksstehende Vergleichswert größer ist
<=	kleiner	wahr, wenn der linksstehende Vergleichswert kleiner oder
>=	größer	wahr, wenn der linksstehende Vergleichswert größer oder

Beispiele für SQL-Befehle zur Auswahl bestimmter Datensätze (Zeilen):

1. Alle Fahrschüler aus Schorndorf.

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

Hinweis: zu vergleichender **Text in Anführungszeichen**

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

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

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

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

4. Alle Fahrschüler, die mehr als 20 Fahrstunden haben.

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

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

5. Alle Fahrschüler, die vor 1998 geboren sind.

```
SELECT *
FROM fahrschueler
WHERE geburtsdatum < '1998-01-01';
```

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

6. … wie Aufgabe 5, zusätzlich nach dem Alter absteigend sortiert.

```
SELECT *
FROM fahrschueler
WHERE geburtsdatum < '1998-01-01'
ORDER BY geburtsdatum ASC;
```

7. 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

8. 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.

9. Alle Fahrschüler die zwischen 1. Januar 1997 und 31. Dezember 1998 geboren sind.

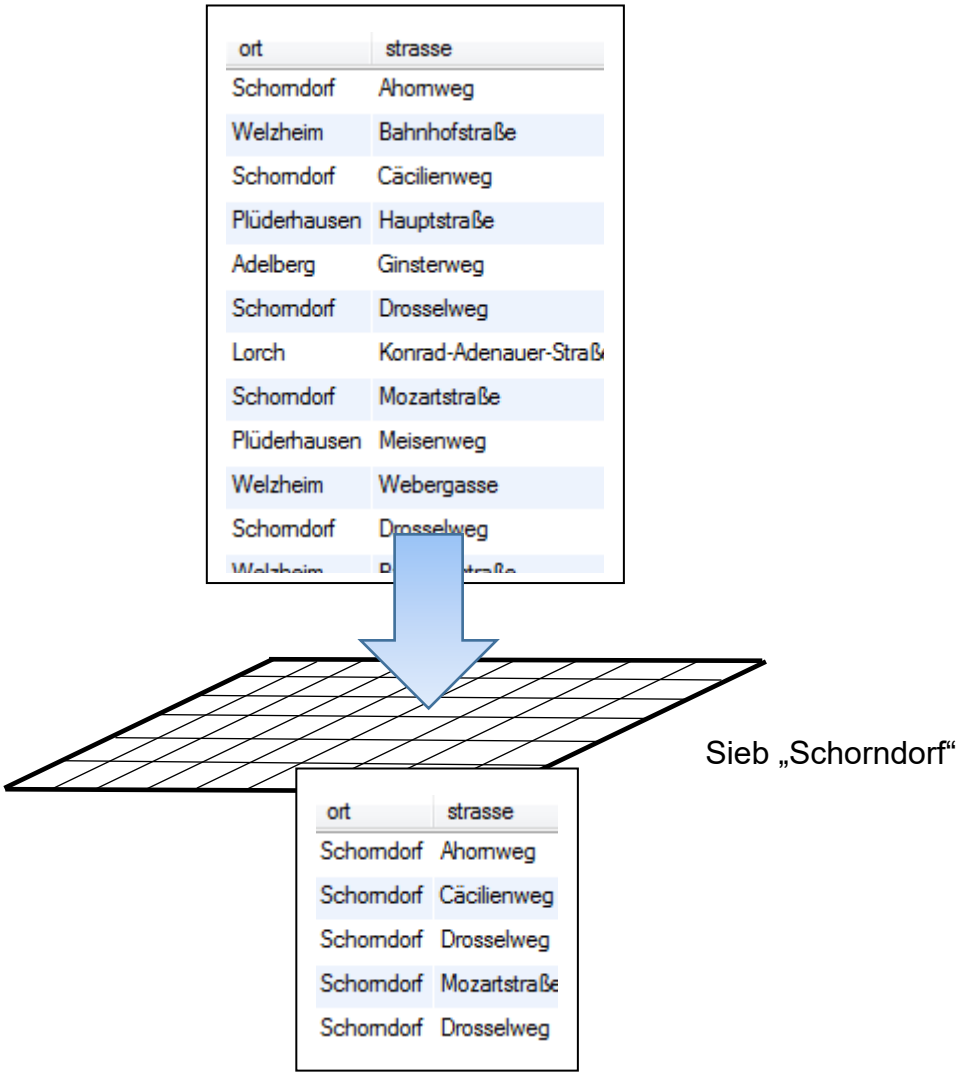
```
SELECT *
FROM fahrschueler
WHERE geburtsdatum >= '1997-01-01'
AND geburtsdatum <= '1998-12-31';
```

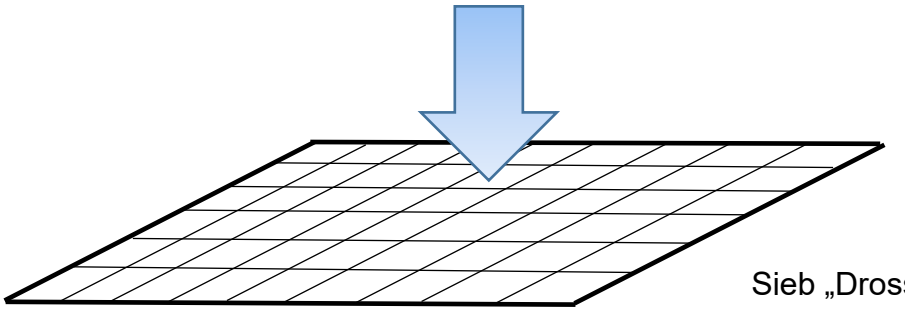
10. Eine Liste, die alle Fahrschüler enthält, die vor dem 01.01.1997 geboren sind, als auch solche, die nach dem 31.12.1998 geboren sind.

```
SELECT *
FROM fahrschueler
WHERE geburtsdatum < '1997-01-01'
OR geburtsdatum > '1998-12-31';
```

11. Alle Fahrschüler, die nicht in Schorndorf wohnen.

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





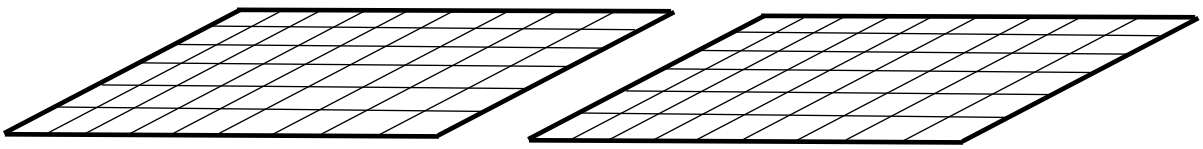
Sieb „Drosselweg“

ort	strasse
Schorndorf	Drosselweg
Schorndorf	Drosselweg

geburtsdatum
1999-12-02
1998-06-20
1995-07-15
1998-11-23
1997-03-24
1999-02-04
1996-05-13
1998-12-24
1997-09-20
1998-12-23
1997-10-22
1998-07-30

Sieb „geboren vor 1997“

Sieb „geboren nach 1998“



geburtsdatum
1995-07-15
1996-05-13

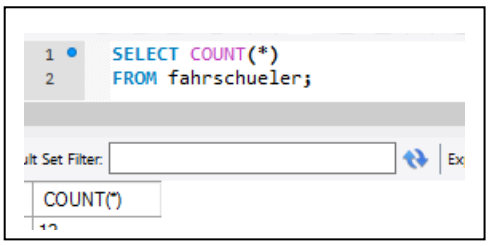
geburtsdatum
1999-12-02
1999-02-04

5.1.4.4 Daten mit SQL abfragen - Beispiele für SQL-Befehle, die mit Funktionen arbeiten

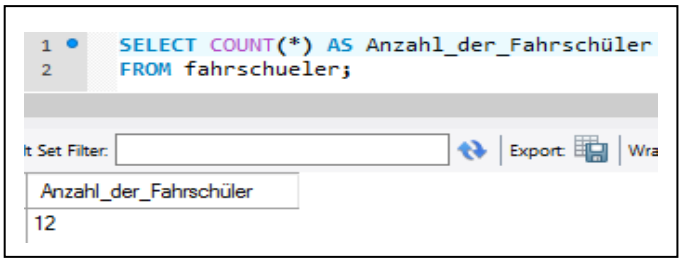
Funktionen können verwendet werden, um

- die Anzahl der 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*)

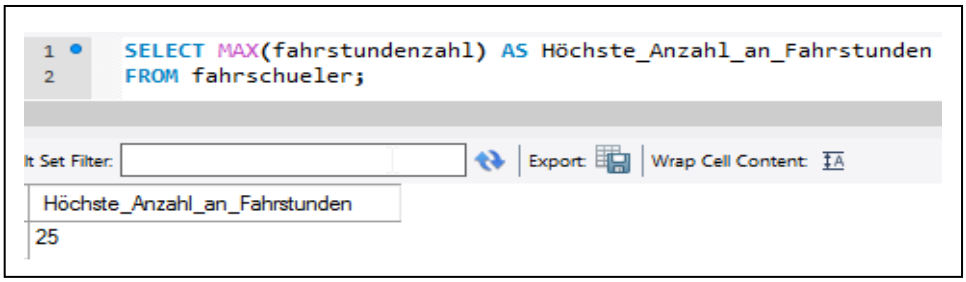
1. Anzahl aller Fahrschüler.



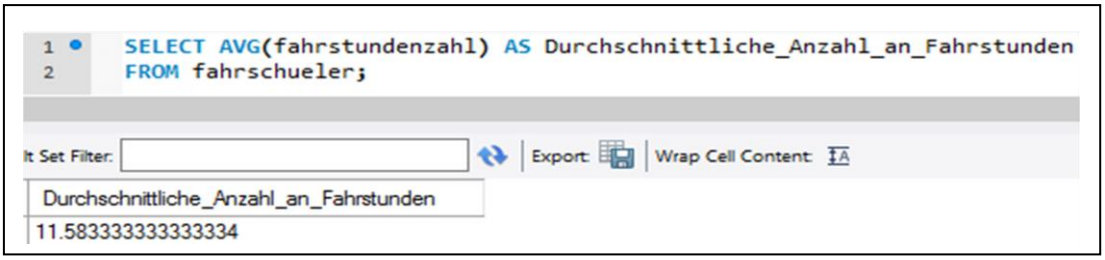
Es ist sinnvoll, für das Funktionsergebnis einen Alias (Andersnamen) zu verwenden. Enthält der Alias Leerzeichen, so ist er in Hochkommas zu setzen.



2. Wie hoch ist die höchste Anzahl an Fahrstunden?



3. Wie hoch ist die durchschnittliche Anzahl an Fahrstunden?



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;
```

lt Set Filter: Export: Wrap Cell Content:

Durchschnittliche_Anzahl_an_Fahrstunden
11.58

4. Wie viele Fahrstunden wurden insgesamt durchgeführt?

```
SELECT SUM(fahrstundenzahl) AS Summe_der_Fahrstunden
FROM fahrschueler;
```

lt Set Filter: Export: Wrap Cell Content:

Summe_der_Fahrstunden
139

Beispiele für SQL-Befehle, die Berechnungen enthalten:

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

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

lt Set Filter: Export: Wrap Cell Content:

nachname	vorname	Umsatz_aus_Fahrstunden_in_€
Abele	Andreas	60
Beutel	Barbara	360
Cramer	Conradt	90
Deiß	Dagmar	420
Emmrich	Anton	180
Dressel	Dagmar	30
Fernandes	Enrico	450
Lutz	Frederik	750
Grund	Dietmar	390
Demürel	Ali	690
Dressel	Andreas	90
Schmidt	Alex	660

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

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

Export: Wrap Cell Content:

Umsatzsumme_aus_Fahrstunden_in_€
4170

```
SELECT SUM(fahrstundenzahl * 30)  AS 'Umsatzsumme aus Fahrstunden in €'
FROM fahrschueler;
```

Export: Wrap Cell Content:

Umsatzsumme aus Fahrstunden in €
4260

5.1.4.5 Daten mit SQL abfragen - Group by

Die **GROUP BY** – Klausel fasst Datensätze, die dieselben Werte enthalten, zu einer Zeile zusammen.

GROUP BY <u>ohne</u> Funktionen

1. Liste aller Orte mit Postleitzahl. Gleiche Orte werden nur einmal angezeigt.

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

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	plz
Adelberg	73099
Lorch	73547
Plüderhausen	73655
Schorndorf	73614
Welzheim	73642

Dabei wird automatisch nach ort aufsteigend sortiert.

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

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

Hier wird nach plz aufsteigend sortiert, weil nach plz gruppiert wird. Man kann Doppel- oder Mehrfachnennungen auch mit dem Befehlswort DISTINCT verhindern:

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

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

Die Reihenfolge wird hier von der Anordnung der Datensätze in der Ausgangstabelle bestimmt.

ABER: Man darf hinter SELECT nur dann Funktionen in Kombination mit Attributen verwenden, wenn man von diesen Attributen Gruppen bildet mit der GROUP BY - Klausel!

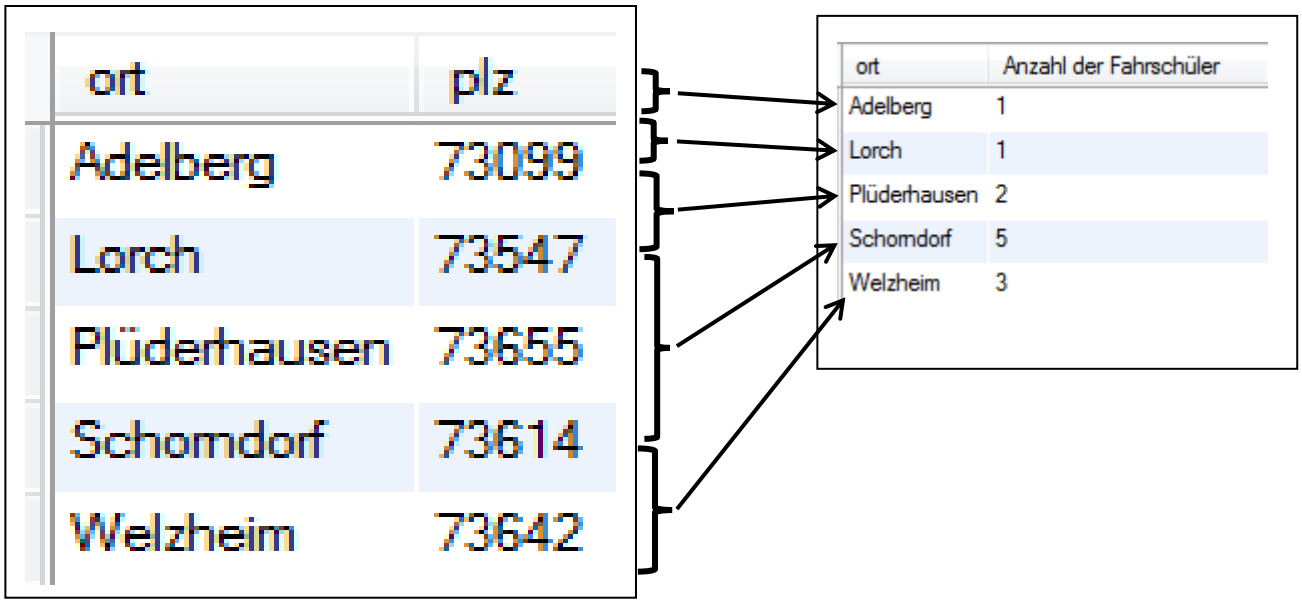
GROUP BY mit Funktionen

2. Anzahl der Fahrschüler je Wohnort.

```
SELECT ort, COUNT(*) AS 'Anzahl der Fahrschüler'
FROM fahrschueler
GROUP BY ort;
```

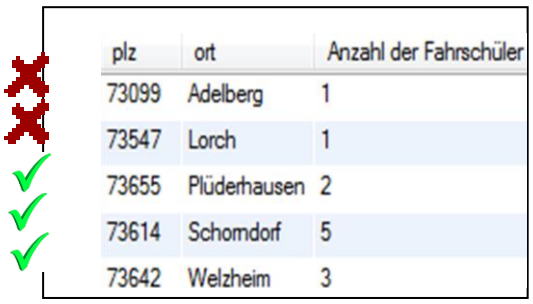
Die **GROUP BY** – Klausel kann mit der **HAVING** – Klausel erweitert werden.

Mit der **HAVING** - Klausel wird festgelegt, welche der gruppierten Sätze angezeigt werden sollen. HAVING wirkt also nur zusammen mit GROUP BY und zeigt alle von GROUP BY gruppierten Datensätze an, die die HAVING-Bedingung erfüllen (HAVING wirkt also ähnlich wie WHERE).



3. Anzahl der Fahrschüler je Wohnort, deren PLZ mit 736 beginnt.

```
SELECT plz, ort, COUNT(*) AS 'Anzahl
der Fahrschüler'
FROM fahrschueler
GROUP BY ort
HAVING plz LIKE '736%';
```



plz	ort	Anzahl der Fahrschüler
73099	Adelberg	1
73547	Lorch	1
73655	Plüderhausen	2
73614	Schorndorf	5
73642	Welzheim	3

4. Anzahl der Fahrschüler mit der gleichen Anzahl an Fahrstunden, deren Fahrstundenanzahl über 2 beträgt.

```
SELECT fahrstundenzahl, COUNT(*) AS 'Anzahl der Fahrschüler'
FROM fahrschueler
GROUP BY fahrstundenzahl
HAVING fahrstundenzahl>2;
```

5.1.5.1 Daten mit SQL verwalten - Daten einfügen

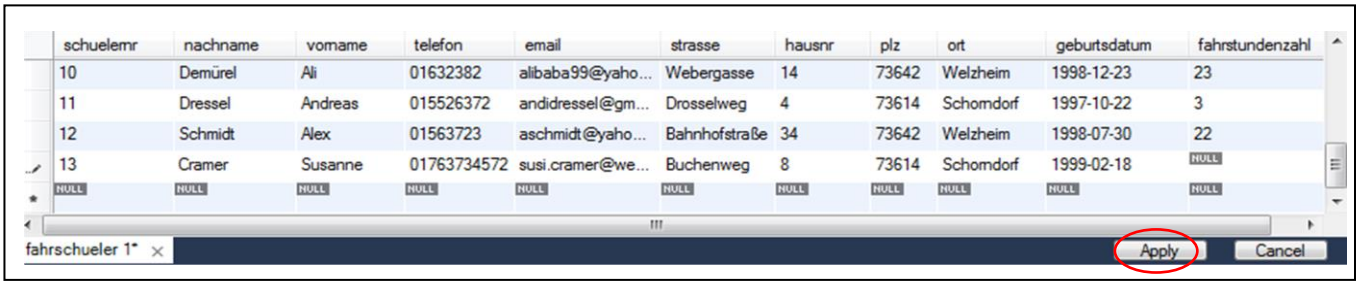
In der Zwischenzeit hat sich die Fahrschule Blechle einen guten Ruf erworben, und es melden sich neue Fahrschüler an.

Folgende neue Fahrschülerin soll in der Tabelle **fahrschueler** erfasst werden:

- Susanne Cramer, wohnhaft in 73614 Schorndorf, Buchenweg 8, Telefon: 01763734572, E-Mail: susi.cramer@web.de, geboren am 18.02.1999

Die neue Fahrschülerin wird an die Liste angefügt, die aufgrund des SQL-Befehles erscheint:

```
Select * from fahrschueler;
```



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

Hinweise:

- Wir lassen den Namen der aktuellen Datenbank weg, und wir lassen die Hochkommata bei den Datenbanknamen, Tabellennamen und Attributnamen weg.
- Wir lassen die Hochkommata bei Attributwerten mit numerischem Datentyp (INT bei schuelernr, DOUBLE bei fahrstundenzahl) weg.

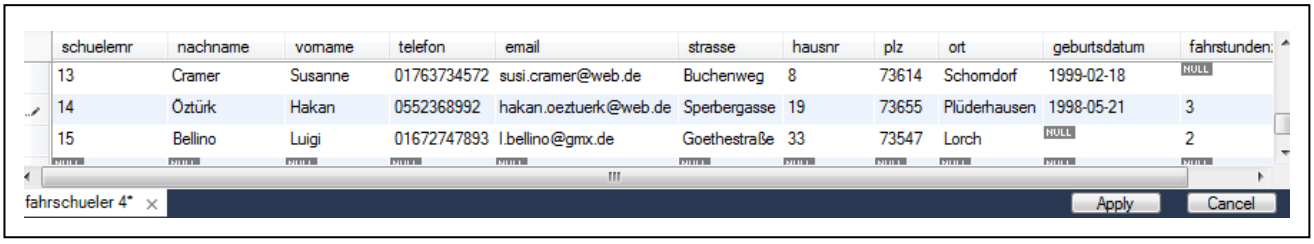
Erläuterung:

INSERT INTO fahrschueler (schuelernr, nachname, vorname, telefon, email, strasse, hausnr, plz, ort, geburtsdatum) VALUES (13, 'Cramer', 'Susanne', '01763734572', 'susi.cramer@web.de', 'Buchenweg', '8', '73614', 'Schorndorf', '1999-02-18');
Füge einen Datensatz ein in der Tabelle fahrschueler in die folgenden Attribute (schuelernr, nachname, vorname, telefon, email, strasse, hausnr, plz, ort, geburtsdatum) mit den Werten VALUES (13, 'Cramer', 'Susanne', '01763734572', 'susi.cramer@web.de', 'Buchenweg', '8', '73614', 'Schorndorf', '1999-02-18');

Achtung: Reihenfolge der Attribute muss gleich der Reihenfolge der Werte sein!

5.1.5.2 Daten mit SQL verwalten - Daten ändern

Von Hakan Öztürk sind bisher nur Nachname, Vorname, Telefon, Straße, Hausnummer, PLZ und der Ort bekannt (siehe Tabelle). In deiner Datenbank fehlt der Datensatz von Hakan aber noch. Füge nun Hakans Datensatz, aber lediglich mit den oben aufgezählten Daten, hinzu. Im Laufe der Zeit teilt dir Hakan dann noch seine Emailadresse und sein Geburtsdatum mit: hakan.oeztuerk@web.de, 21.Mai 1998. Mittlerweile hat er 3 Fahrstunden absolviert.




```
UPDATE `fahrschule`.`fahrschueler`
SET `email`='hakan.oeztuerk@web.de', `geburtsdatum`='1998-05-21', `fahrstundenzahl`='3'
WHERE `schuelernr`='14';
```

Erläuterung:

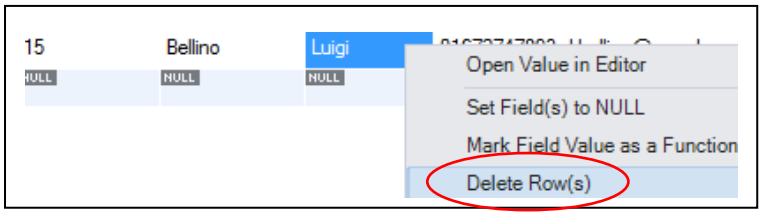
UPDATE fahrschule.fahrschueler SET email = 'hakan.oeztuerk@web.de', geburtsdatum = '1998-05-21', fahrstundenzahl = 3 WHERE schuelernr = 14;
Aktualisiere die Tabelle fahrschueler setze email = 'hakan.oeztuerk@web.de', geburtsdatum = '1998-05-21', fahrstundenzahl = 3 bei dem Datensatz mit der schuelernr 14;

Hinweis:

Aus Sicherheitsgründen ist die Workbench standardmäßig so konfiguriert, dass **UPDATE**-Befehle nur mit dem Primärschlüssel in der **WHERE**-Klausel durchgeführt werden.

5.1.5.3 Daten mit SQL verwalten - Daten löschen

Luigi Bellino möchte nun doch lieber eine Fahrschule in seinem Wohnort besuchen. Deshalb wird dieser Datensatz nicht mehr benötigt und soll gelöscht werden.




```
DELETE FROM `fahrschule`.`fahrschueler` WHERE `schuelernr`='15'
```

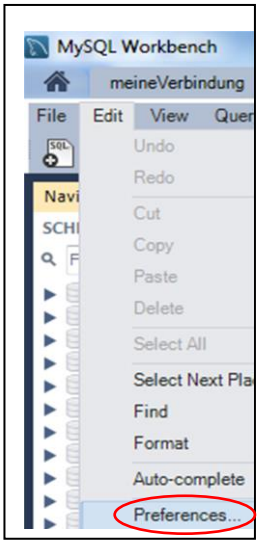
Erläuterung:

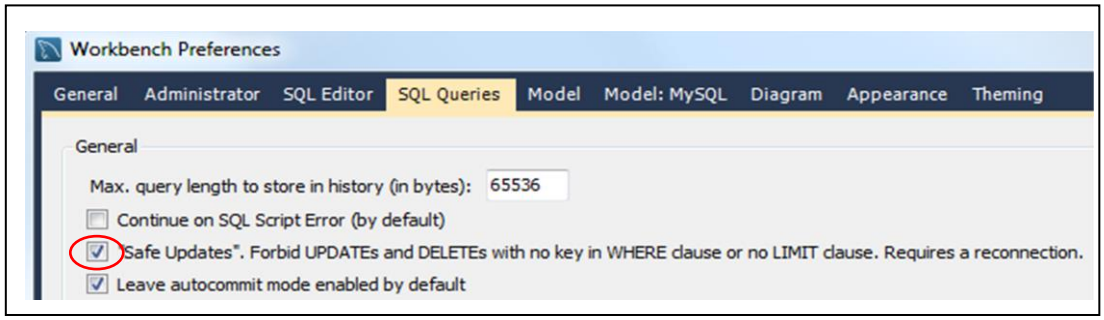
DELETE FROM fahrschule.fahrschueler WHERE schuelernr = 15;
Lösche aus der Tabelle fahrschueler den Datensatz mit der schuelernr 15;

Hinweis:

Aus Sicherheitsgründen ist die Workbench standardmäßig so konfiguriert, dass **DELETE**-Befehle nur mit dem Primärschlüssel in der WHERE-Klausel durchgeführt werden.

Diese Einstellung kann so geändert werden:





Nach dieser Änderung (Haken entfernen oder hinzufügen) muss die aktuelle Verbindung geschlossen und wieder erneut geöffnet werden (reconnect).