Relationale Datenbanken:

SQL

Inhaltsverzeichnis:

1	SQL	2
1.1	SQL als Data Manipulation Language (DML)	3
1.1.1	Datensätze in Tabellen hinzufügen mit dem INSERT INTO- Befehl	4
1.1.2	Datensätze ändern mit dem UPDATE- Befehl (Aktualisierungsabfragen)	10
1.1.3	Datensätze löschen	13
1.2	SQL als Query Language (QL)	15
1.2.1	Einführung	15
1.2.2	Auswahlabfragen am Beispiel der Tabelle Personal	16
1.2.3	Lösungshinweise zu Auswahlabfragen am Beispiel der Tabelle Personal	18
1.2.4	Übersicht der Hauptklauseln der SELECT-Anweisung	22
1.2.5	Auswahlabfragen mit mehreren Tabellen	23
1.2.6	Datenbankview	26
1.2.7	Auswahlabfragen mit Gruppierung	33
1.2.8	Auswahlabfragen mit Unterabfragen	36
1.3	SQL und Boolesche Algebra	39

1 SQL (Structured Query Language)

Zum Bearbeiten von Datenbanken wird die Standardsprache SQL (Structured Query Language)verwendet. Die meisten heute verbreiteten Datenbanksysteme implementieren Teile des SQL-Sprachstandards. Dadurch ist es möglich, Anwendungsprogramme zu erstellen, die vom verwendeten Datenbanksystem unabhängig sind.

SQL (Structured Query Language) ist eine Datenbanksprache zur Definition, Manipulation und Abfrage von Daten in relationalen Datenbanken .

Datendefinitionen	**Datenmanipulationen**	**Datenbankabfragen** -
Data Definition Language (**DDL**)- Datenbankobjekte (Datenbanken, Tabellen) erstellen, ändern und löschen.	Data Manipulation Language (**DML**) Daten manipulieren (einfügen , ändern, und löschen)	Data Query Language (**DQL**) Daten in der Datenbank auswählen, aufbereiten und auslesen
Datenkontrollen - Data Control Language (DCL) Überwachung der Datenbank und der Operationen mit der Datenbank. Sie wird verwendet, um Berechtigungen zu vergeben oder zu entziehen.		

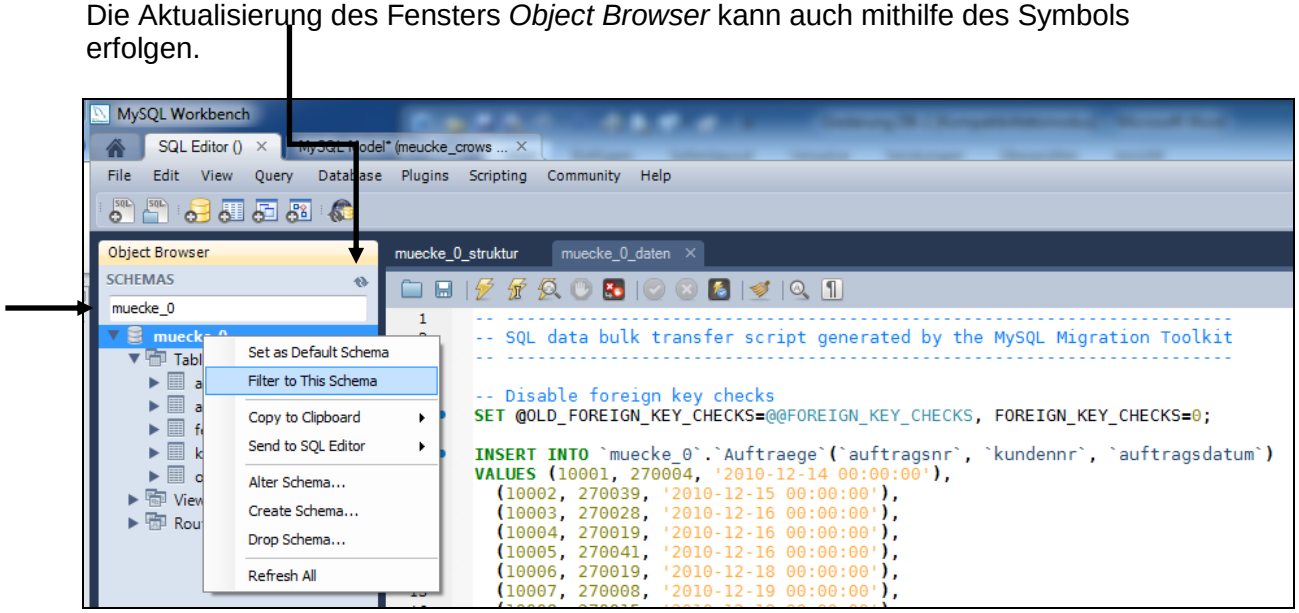
1.1 SQL als Data Manipulation Language (DML)

Vorbemerkung:

Grundlage für dieses Kapitel „SQL-Abfragen“ ist die Datenbank *muecke_0*. Um mit der Datenbank arbeiten zu können, müssen nacheinander das Skript *muecke_0_struktur*.sql für die Struktur der Datenbank und das Skript *muecke_0_daten.sql* mit den Daten geöffnet und ausgeführt werden.

Vorgehensweise

- MySQL-Datenbankserver und die *MySQL Workbench* starten (siehe Seite 76)
- Skripts mit den SQL-Befehlen zur Erstellung der Datenbank um mit den Befehlen zum Einfügen der konkreten Daten öffnen (Menü: *File* → *Open SQL Skript…*)
- Die Skripte nacheinander ausführen (Zuerst das Skript *muecke_0_struktur)*
- Die Anzeigen im Fenster *Object Browser* aktualisieren. Dazu wird das Kontextmenü mit der rechten Maustaste geöffnet und der Befehl *Refresh All* ausgeführt.
  Die Aktualisierung des Fensters *Object Browser* kann auch mithilfe des Symbols erfolgen.

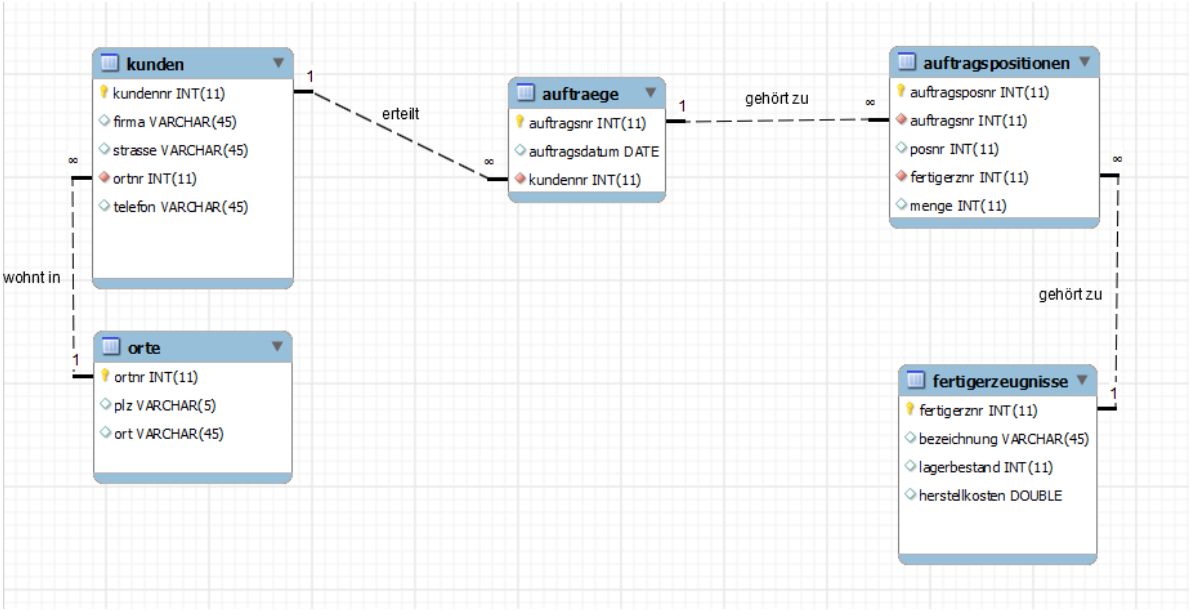


Hinweis: Wenn die zu bearbeitende Datenbank über das Kontextmenü mit dem Menübefehl **`Filter to This Schema`** ausgewählt wird, wird im Object Browser nur noch diese Datenbank mit ihren Objekten angezeigt. Wird der Menübefehl **`Set as Default Schema`** gewählt, ist die Datenbank geöffnet und alle Anweisungen beziehen sich auf diese Datenbank.

.

1.1.1 Datensätze in Tabellen mit dem INSERT INTO-Befehl hinzufügen

Das nachfolgende erweiterte Entity Relationship Model (EERM) zeigt die Tabellen der Datenbank muecke_0 mit ihren Beziehungen und Beziehungskardinalitäten.



In der folgenden Übersicht sind die Beziehungen zwischen den Tabellen und deren Auswirkungen auf die Datenintegrität beim Erfassen beziehungsweise Löschen von Daten aus der Datenbank beschrieben.

Tabelle	Beschreibung der Beziehung	Auswirkung der referentiellen Integrität
orte	Ein Ort kann Wohnsitz von einem oder mehreren Kunden sein.	Ein Wohnort kann aus der Tabelle *orte* nicht gelöscht werden, wenn Kunden mit dem Wohnort in der Tabelle *kunden* enthalten sind.
kunden	Ein Kunde hat einen ersten Wohnsitz.	In der Tabelle *kunden* kann eine *ortnr* nur erfasst werden, wenn diese in der Tabelle *orte* vorhanden ist.
	Ein Kunde kann einen oder mehrere Aufträge erteilen.	Ein Kunde kann aus der Tabelle *kunden* nicht gelöscht werden, wenn es in der Tabelle *auftraege* noch Aufträge dieses Kunden gibt.
auftraege	Ein Auftrag wird von einem Kunden erteilt.	In der Tabelle *auftraege* kann der Auftrag eines Kunden erst erfasst werden, wenn die Kundennummer (Attribut *kundennr*) in der Tabelle *kunden* vorhanden ist.
	Ein Auftrag enthält eine oder mehrere Positionen.	Ein Auftrag kann aus der Tabelle *auftraege* nicht gelöscht werden, wenn es in der Tabelle *auftragspositionen* noch Positionen mit der Auftragsnummer dieses Auftrages gibt.
auftrags-positionen	Eine Auftragsposition gehört zu einem Auftrag.	Eine Auftragsposition kann in der Tabelle *auftragspositionen* erst erfasst werden, wenn der Auftrag mit der entsprechenden Auftragsnummer (Attribut *auftragsnr*) in der Tabelle *auftraege* angelegt ist.
	Eine Auftragsposition bezieht sich auf ein Fertigerzeugnis.	In der Tabelle *auftragspositionen* kann die Eingabe eines Fertigerzeugnisses (Attribut *fertigerznr*) erst erfolgen, wenn die Fertigerzeugnissnummer (Attribut *fertigerznr*) in der Tabelle *fertigerzeugnisse* vorkommt.
fertiger-zeugnisse	Ein Fertigerzeugnis kann in einer oder mehreren Auftragspositionen enthalten sein.	Ein Fertigerzeugnis kann aus der Tabelle *fertigerzeugnisse* nicht gelöscht werden, wenn dieses noch in der Tabelle *auftragspositionen* vorkommt (Attribut *fertigerznr*).

Um einen Widerspruch zu den Anforderungen der *referentiellen Integrität* beim Erfassen der Daten zu vermeiden, müssen also zunächst die Daten der Parent-Tabellen und danach die Daten der Child-Tabellen eingegeben werden.

Übersicht über die Reihenfolge beim Erfassen von Daten in die Datenbank *muecke_0*:

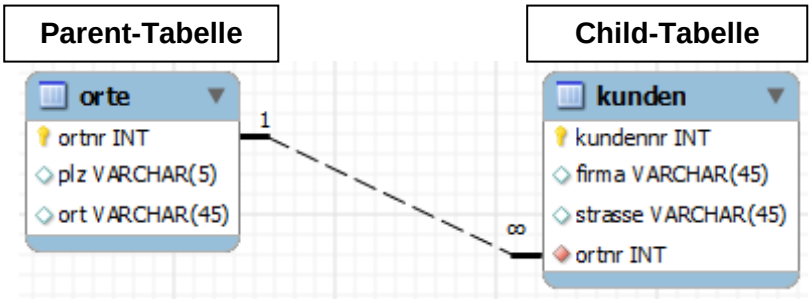
	Parent-Tabelle	**Primärschlüssel**	**Child-Tabelle**	**Fremdschlüssel**
1.	orte	ortnr		
2.	kunden	kundennr	kunden	ortnr
3.	auftraege	auftragsnr	auftraege	kundennr
5.	fertigerzeugnisse	fertigerznr		
6.	auftragspositionen	auftragsposnr	auftragspositionen	auftragsnr fertigerznr

Beispiel 1: Daten in die *Tabelle* kunden erfassen - ohne Verstoß gegen die referentielle Integrität

Situation: Die Firma Möbel - Kaiser aus 81241 München, Theresienwiese 12, Telefonnummer 089-223520, wird als Kunde geworben und erhält die Kundennummer 270056. In der Tabelle *orte* hat der Eintrag des Ortes 81241 München die *ortnummer* 33011.
Diese Daten sollen in der Tabelle *kunden* erfasst werden.

Erläuterung

Die aufzunehmenden Kundendaten betreffen die Tabellen *kunden* und *orte.*
Zwischen den beiden Tabellen besteht eine 1 : N - Beziehung (siehe nachfolgende Abbildung). Die Beziehung wird durch die Verknüpfung von Schlüsselattributen erzeugt. Hierzu muss auf der N-Seite, in der Child-Tabelle *kunden,* der Primärschlüssel der Parent-Tabelle *orte* als Fremdschlüssel eingefügt werden.



Die Aufnahme eines Kunden in die Child–Tabelle *kunden* setzt voraus, dass der Attributwert, auf den sich der Fremdschlüssel *ortnr* bezieht, in der Parent-Tabelle *orte*, vorhanden ist (= referentielle Integrität).

Child-Tabelle	**Primärschlüssel**	**Fremdschlüssel**	**Parent-Tabelle**	**Primärschlüssel**
kunden	kundennr	**ortnr**	orte	**ortnr**

Tabelle *kunden (=Child)* **Tabelle *orte (=Parent)***

kundennr	firma	strasse	ortnr	telefon
270053	Einrichtungshaus Bluthardt	Kastellstr. 14	29303	0711-457868
270055	Möbel Schmidt GmbH & Co KG	Friedhofsweg 1	38490	0931-547780
270056	Möbel - Kaiser	Theresienwies..	33019	089-223520

ortnr	plz	ort
33017	81237	München
33018	81238	München
33019	81241	München

Vorgehensweise bei der Datenerfassung

1 Prüfen, ob die Referenzdaten in der Parent-Tabelle *orte* vorhanden sind:

SQL-Anweisung

```
SELECT *
FROM orte
WHERE plz="81241";
```

Ergebnis

ortnr	plz	ort
33019	81241	München
NULL	NULL	NULL

2 Daten in die Tabelle kunden erfassen:

Die *INSERT INTO* - Anweisung

Syntax: INSERT INTO <*tabellenname*>(<*attributbezeichnung1*>, <*attributs-bezeichung2*>, …) VALUES (<*attributwert1*>, <*attributwert2*>, …);

Erläuterungen:

Absicht	**Befehl**
Text einfügen	INSERT... (... Name, ...) VALUES (...“Knüller“, ...)
Datumswert einfügen	INSERT... (... Lieferdatum, ...) VALUES (..."2005-07-19", ...)
numerischer Wert einfügen	INSERT... (... Umsatz, ...) VALUES (... 4000.90, ...)
Berechnung einfügen	INSERT... (... Bruttowert, ...) VALUES (... 280*1.16, ...)

Hinweis: Einige SQL-Dialekte verlangen folgende Schreibweise für Datumsangaben: …#2005-07-19# …

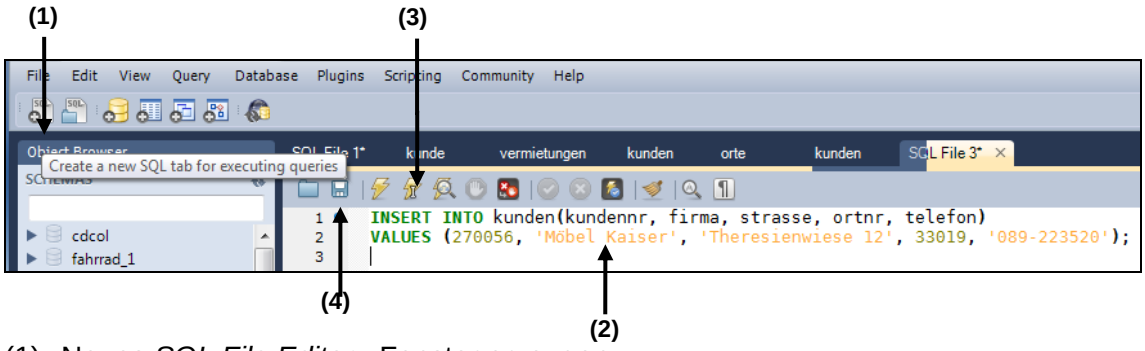
SQL-Anweisung:

```
INSERT INTO kunden(kundennr, firma, strasse, ortnr, telefon)
VALUES (270056, 'Möbel Kaiser', 'Theresienwiese 12', 33019, '089-223520');
```

3 Datenerfassung mit der MySQL Workbench

SQL-Anweisungen können in der *MySQL Workbench* mit Hilfe des SQL File Editors oder im Dialog mit Unterstützung des Programmassistenten eingegeben werden.

- **Eingabe mit dem** *SQL File Editor*



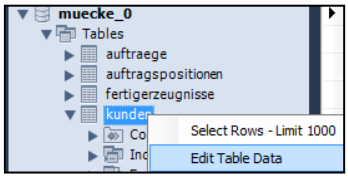
(1) Neues *SQL File Editor* - Fenster erzeugen.

(2) SQL-Anweisung erfassen.

(3) SQL-Anweisung ausführen. Im „Output"-Fenster erscheint eine Meldung zur Bestätigung:

(4) Die SQL-Anweisung kann mit dem Menübefehl **`File→Save Script As…`** gespeichert und später wieder aufgerufen werden.

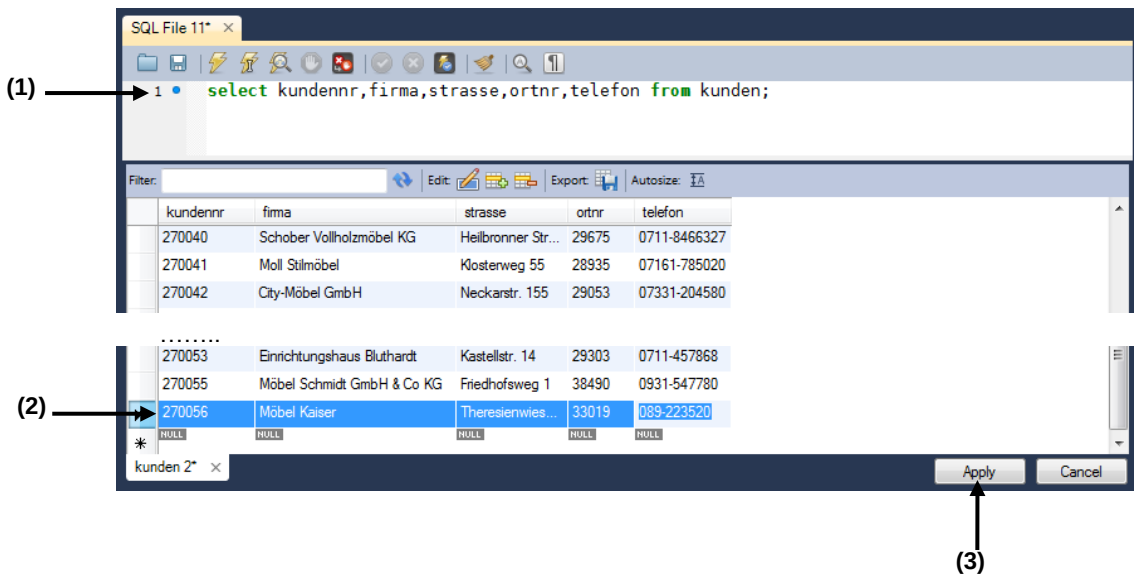
- **Eingabe im Dialog mit dem Programmassistenten**



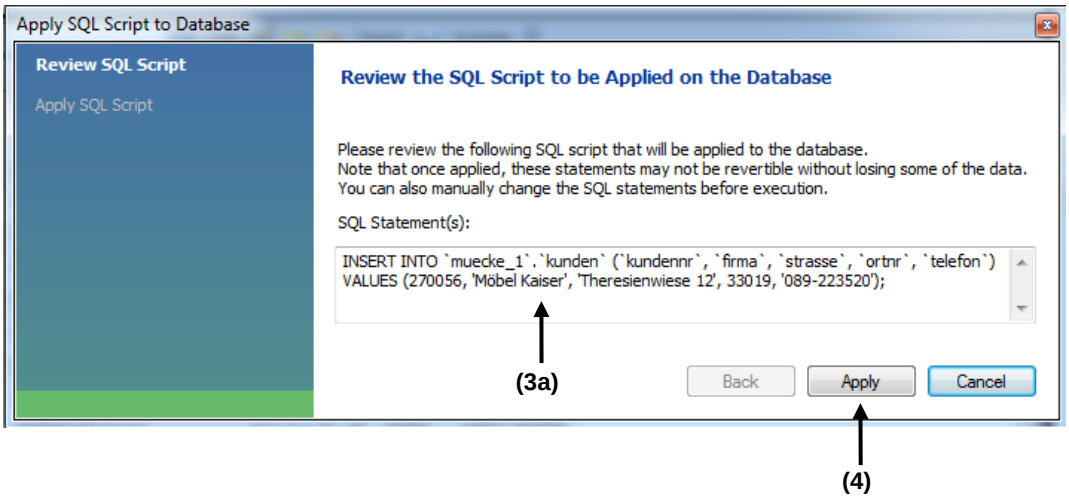
Bei dieser Vorgehensweise

(1) listet man zunächst alle Datensätze auf (Kontextmenü zum Tabellennamen *kunden* im Fenster *Objektct Browser* und

(2) fügt dann am Ende der Liste die Daten des zu erfassenden Kunden hinzu.



(3) Wenn dann die Befehlsschaltfläche [**Apply**] gedrückt wird, wird die SQL-Anweisung erzeugt (3a).

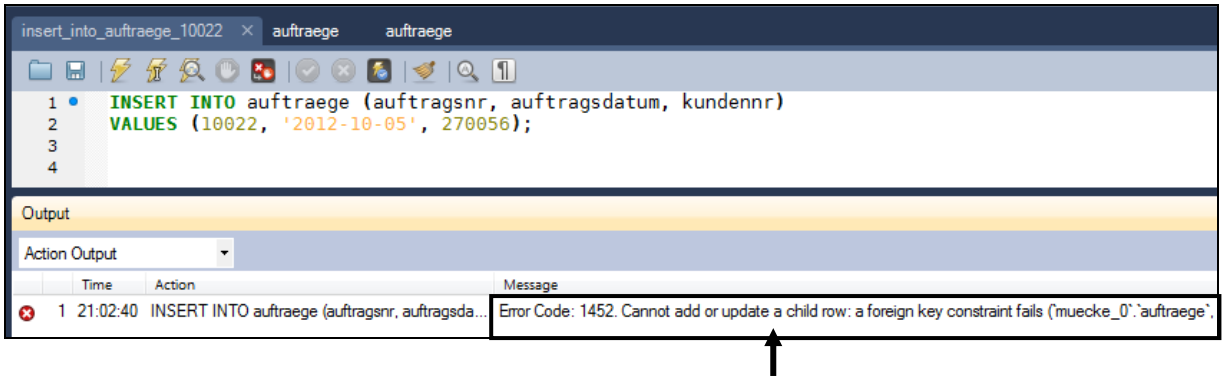


(4) Mit einem weiteren Click auf die Befehlsschaltfläche **[Apply]** und anschließend auf **[Finish]** wird die Anweisung auf dem Datenbankserver ausgeführt.

Beispiel 2: Daten in die *Datenbank* muecke_0 *mit Verstoß* gegen die referentielle Integrität erfassen.

Situation: Herr Klaus Meinhardt, Inhaber der Firma "Meinhardt - Möbel nach Maß", 70599 Stuttgart, Neckarstraße 55, Telefon 0711-453423 bestellt am 05.10.d.J. auf der Hausmesse der Mücke & Partner GmbH spontan 9 Sideboards Donau (*fertigerznr 200010*) und 2 Sideboards Ultimo *(fertigerznr 200016*).

Ohne zu prüfen, ob der Kunde "Meinhardt - Möbel nach Maß" in der Datenbank erfasst ist, versucht der Verkäufer Franz Schlauch, die Auftragsdaten in der Tabelle *auftraege* zu erfassen und erhält die nachfolgende Meldung:

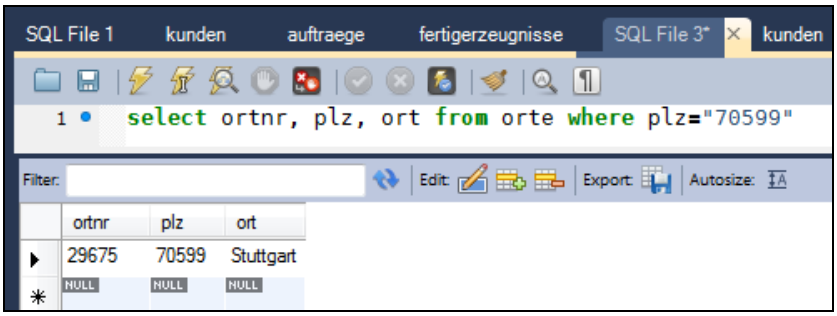


Wie die Übersicht auf der Seite 99 zeigt, kann ein Auftrag in die Tabelle auftraege erst dann erfasst werden, wenn die benötigte Kundennummer (Attribut *kundennr*) in der Tabelle kunden vorhanden ist.

Der Auftrag wird daher wie auf Seite 50 beschrieben, in mehreren Schritten erfasst.

- Erfassen des neuen Kunden mit Hilfe des SQL File Editors.

  (1) Es muss zunächst überprüft werden, ob der Wohnort 70599 Stuttgart in der Tabelle *orte* vorhanden ist.



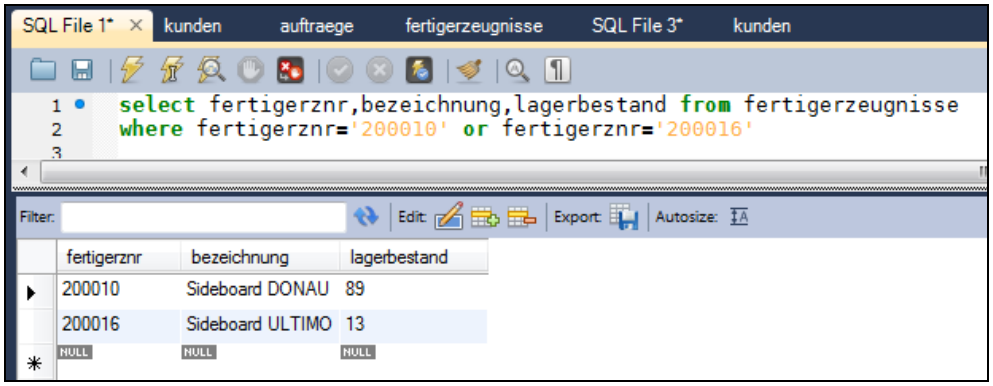
  (2) Erfassen der Kundendaten in die Tabelle *kunden*.

  INSERT INTO kunden (kundennr, firma, strasse, ortnr, telefon) VALUES (270056, 'Meinhardt-Möbel nach Maß', 'Neckarstraße 55', 29675, '0711-453423');

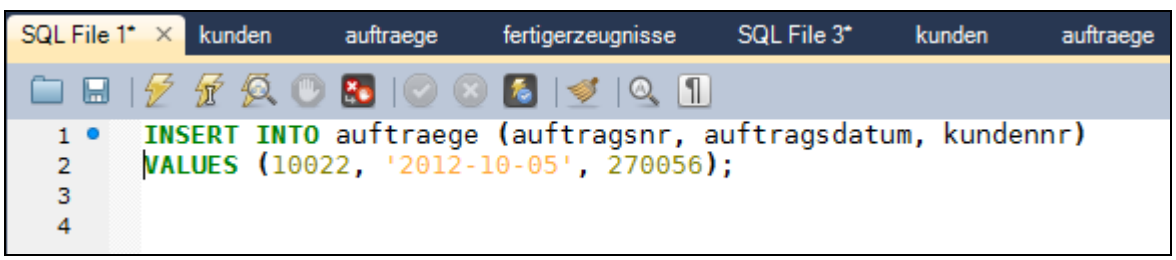
  Die Anweisung wird wie auf den Seiten 7 beschrieben
  - im ***SQL File Editor*** eingegeben oder
  - mit Hilfe der MySQL Workbench generiert

  und ausgeführt.

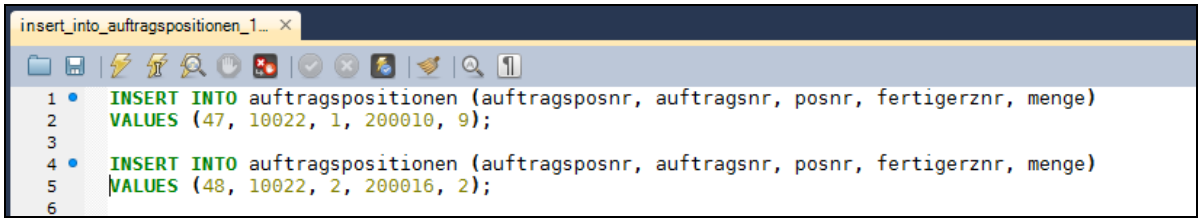
- Prüfen, ob die Fertigerzeugnisse vorrätig sind.



- Auftragskopfdaten in die Tabelle *auftraege* erfassen. Die nächste Auftragsnummer ist 10022.



- Auftragspositionsdaten des Auftrages mit der Auftragsnummer 10022 in die Tabelle *auftragspositionen* erfassen. Die Auftragspositionsnummern sind 47 und 48.



1.1.2 Datensätze mit dem UPDATE- Befehl ändern Aktualisierungsabfragen)

Situation: Wegen gestiegener Kosten sollen bei allen Fertigerzeugnissen die Herstellkosten um 5 % erhöht werden. In der Tabelle *fertigerzeugnisse* sind die Herstellkosten (Attribut *herstellkosten*) durch die erhöhten Herstellkosten zu ersetzen.

In der nachfolgenden Tabelle ist auszugsweise dargestellt, wie die erhöhten Herstellkosten ermittelt werden:

fertigerznr	bezeichnung	lagerbestand	herstellkosten vorher	herstellkosten nachher
200001	Kleiderschrank PARIS	137	1675,5	1759,275
200002	Kleiderschrank LONDON	107	1175,75	1234,5375
200003	Schrankwand ALLGÄU	139	955,5	1003,275
200004	Schrankwand LINZ	108	995	1044,75

1675,5 * 1.05

herstellkosten = herstellkosten * 1.05

Mit dem UPDATE-Statement können ein oder mehrere Datensätze verändert werden. Meistens wird dieses Statement mit einer *WHERE*-Klausel verwendet.

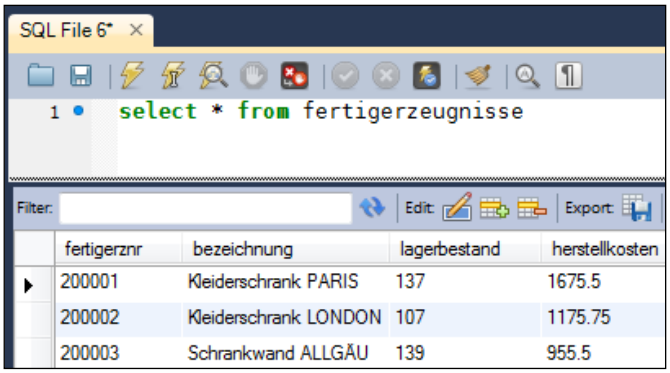
Syntax:	UPDATE <tabellenname> SET <attributbezeichnung1> = <attributswert1>, <attributbezeichnung2> = <attributswert2>, … WHERE <bedingung>;

Beispiele:

Absicht	Befehl
Text ändern	SET name = „Huber“ ...WHERE kurzzeichen = „Arm“
Mehrere Felder ändern	SET name = „Huber“, vorname =“Alex“ ...WHERE kurzzeichen = „Arm
Datumswert ändern	SET lieferdatum = "2013-02-18" ...
Tagesdatum einsetzen	SET bestelldatum = DATE() ...
numerischen Wert ändern	SET umsatz = 7000.75 ...
Feldinhalte löschen	SET umsatz = NULL ...(geht auch bei Datums- und Zeichenfeldern)
mehrere Datensätze ändern	WHERE umsatz < 10000 ... WHERE bestelldatum < "2013-03-22" ...

Die Aktualisierungsabfrage kann wieder mit Hilfe des *SQL File Editors* eingegeben und ausgeführt werden. Zur besseren Kontrolle werden die zu ändernden Datensätze vor und nach der Aktualisierung aufgelistet.

1) Die **zu aktualisierenden Datensätze** auflisten

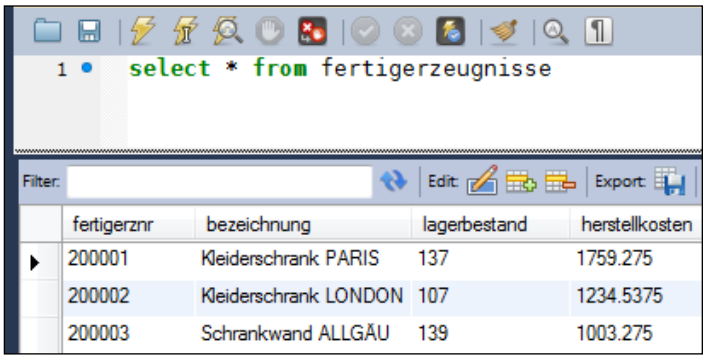


fertigerznr	bezeichnung	lagerbestand	herstellkosten
200001	Kleiderschrank PARIS	137	1675.5
200002	Kleiderschrank LONDON	107	1175.75
200003	Schrankwand ALLGÄU	139	955.5

2) **Aktualisierungsabfrage**

```
UPDATE fertigerzeugnisse
SET herstellkosten = herstellkosten * 1.05;
```

3) Die **aktualisierten Datensätze** auflisten



fertigerznr	bezeichnung	lagerbestand	herstellkosten
200001	Kleiderschrank PARIS	137	1759.275
200002	Kleiderschrank LONDON	107	1234.5375
200003	Schrankwand ALLGÄU	139	1003.275

Hinweis: In der MySQL Workbench ist für den *SQL File Editor* die Option „*Safe Updates*“ aktiviert.

Dadurch wird von einer Aktualisierungs- (*UPDATE*) oder Löschabfrage *(DELETE*) eine WHERE-Klausel mit Angabe eines Schlüssels verlangt.

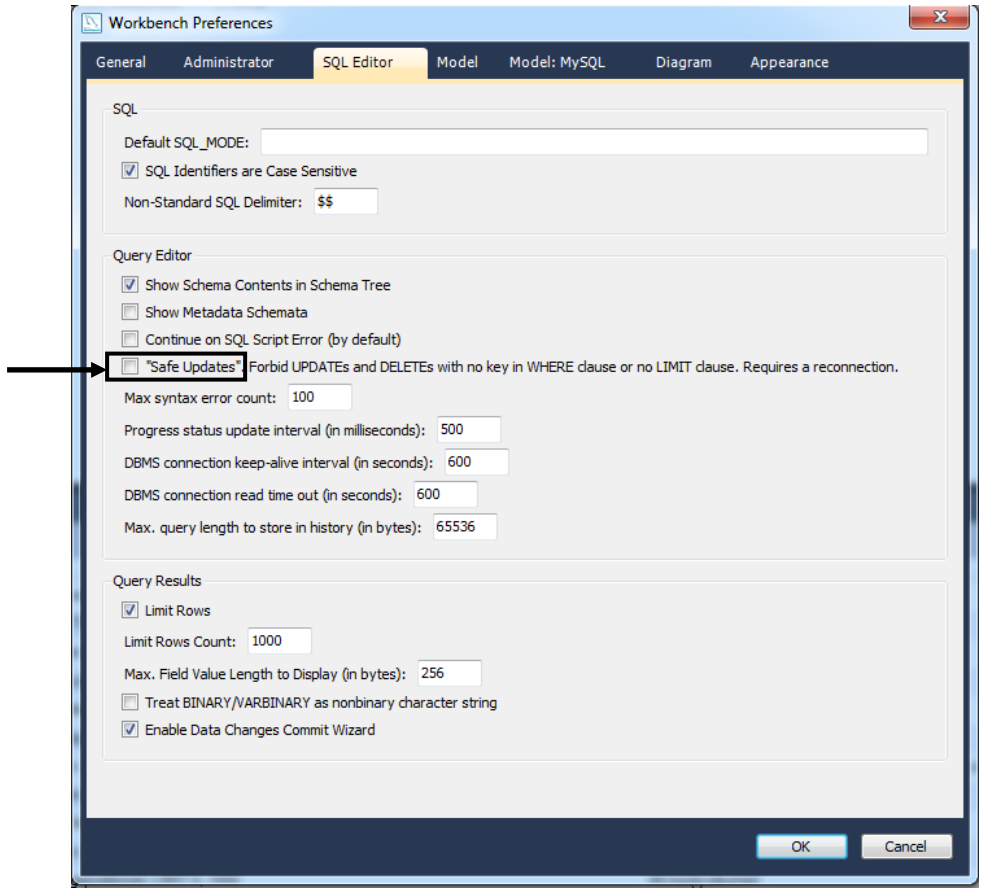
Die SQL-Anweisung ohne WHERE-Klause (siehe oben) löst eine Fehlermeldung aus.

Wird die SQL-Anweisung um die WHERE-Klausel mit Angabe eines Schlüsselwertes erweitert,

```
UPDATE fertigerzeugnisse SET herstellkosten herstellkosten*1.05
WHERE fertigerznr !=0;
```

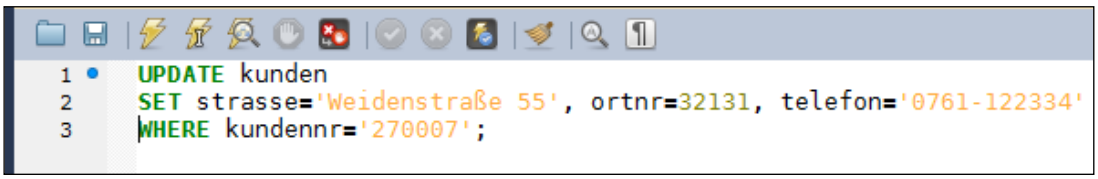
wird sie fehlerfrei ausgeführt.

Diese „Sicherheitseinstellung“ kann mit dem Befehl **`EDIT → Preferences`** durch Deaktivieren der Option „**`Safe Updates`**“ angepasst werden:



Übung: Der Kunde Möbel Bison Heck (Kundennummer 270007) ist umgezogen. Sein Geschäftssitz ist jetzt in der Weidenstraße 55, 70950 Freiburg (ortnr = 32131) Außerdem hat sich die Telefonnummer des Kunden geändert. Neue Nummer: "0761-122334"
Die Datenbank soll um diesen Sachverhalt ergänzt werden.

Lösung:



```
1  UPDATE kunden
2  SET strasse='Weidenstraße 55', ortnr=32131, telefon='0761-122334'
3  WHERE kundennr='270007';
```

1.1.3 Datensätze löschen

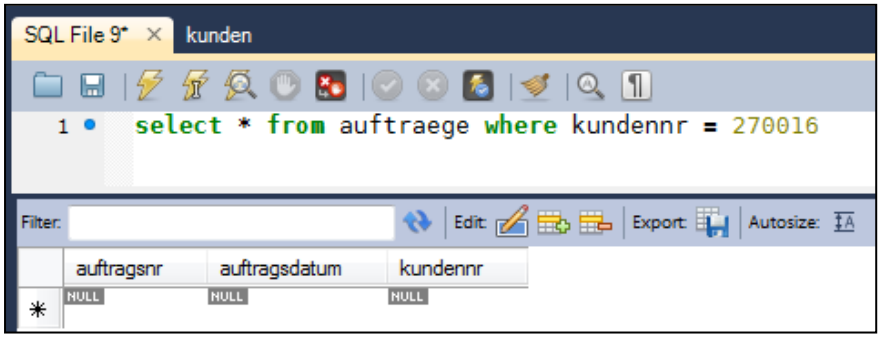
Wie in der Übersicht auf Seite 4 beschrieben, muss beim Löschen von Datensätzen die referentielle Integrität beachtet werden. Datensätze aus Parent-Tabellen können nur gelöscht werden, wenn es keine Verbindungen mehr zu Datensätzen in verknüpften Child-Tabellen gibt.

Beispiel 1: **Löschen eines Kunden - ohne Verletzung der referentiellen Integrität**

Situation: Der Kunde "Wohnwelt GmbH" (Kundennr. 270016) zeigt kein Interesse an einer weiteren Geschäftsbeziehung zur Firma Mücke & Partner OHG. Die Daten dieses Kunden sollen gelöscht werden.

Erklärung: Die Tabelle *kunden* ist Parent-Tabelle zur Tabelle *auftraege*. Das bedeutet, dass ein Kunde nur dann aus der Tabelle *kunden* gelöscht werden kann, wenn es keine Aufträge dieses Kunden mehr in der Tabelle *auftraege* gibt (Siehe Seite4).

- Zuerst wird also überprüft, ob es in der Tabelle *auftraege* noch eine Verbindung zum Kunden Wohnwelt GmbH (kundnummer = 20016) gibt.

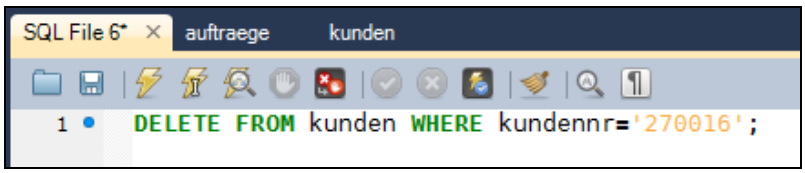


- Löschen des Datensatzes mit Hilfe des SQL File Editors:

  Mit der DELETE-Anweisung können ein oder mehrere Datensätze aus einer Tabelle gelöscht werden.

Syntax:	DELETE FROM <*tabellenname*> WHERE <*bedingung*>;

  Eingabe im SQL File Editor:

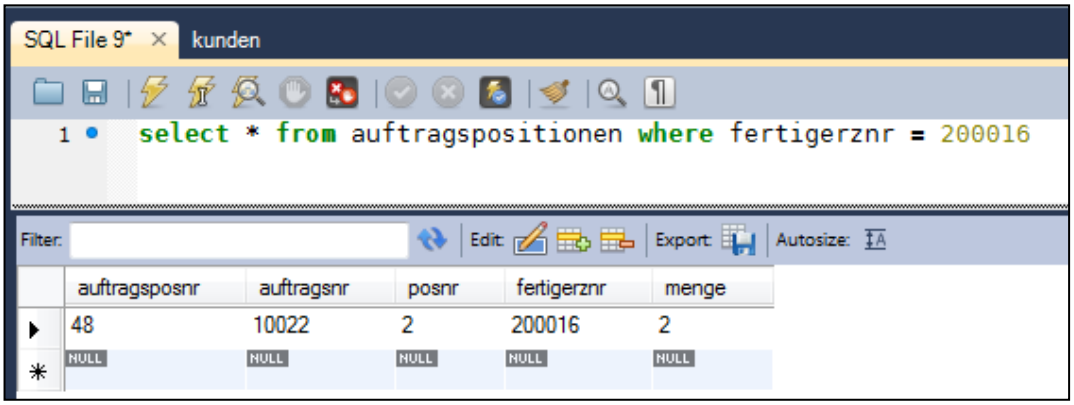


Beispiel 2: **Löschen eines Erzeugnisses - mit Verletzung der referentiellen Integrität**

Situation: Aus der Tabelle *fertigerzeugnisse* soll der Artikel "Sideboard Ultimo" (*fertigerznr = '200016'*) gelöscht werden. Er passt nicht mehr in die Produktstrategie des Unternehmens.

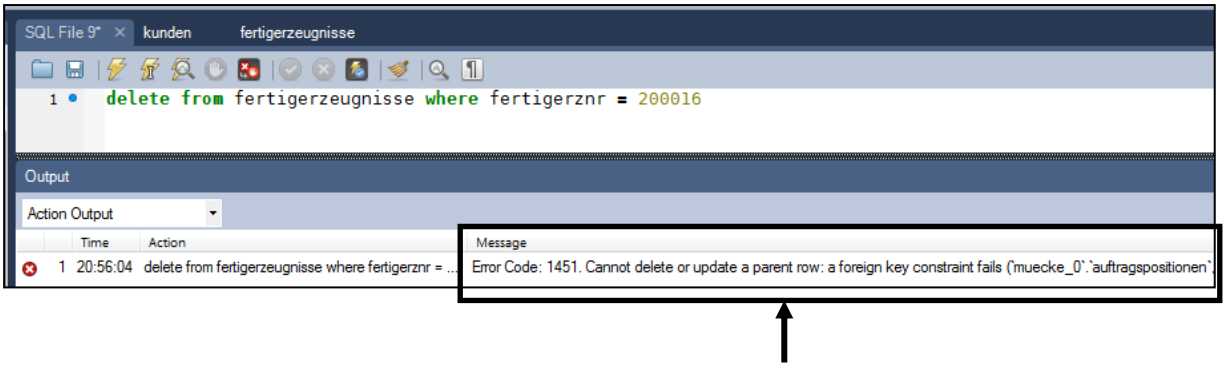
Erklärung Die Tabelle *fertigerzeugnisse* ist Parent-Tabelle zur Tabelle *auftraegs-positionen*. Das bedeutet, dass ein Erzeugnis nur dann aus der Tabelle *fertigerzeugnisse* gelöscht werden kann, wenn es in keiner Auftragsposition der Tabelle *auftragspositionen* mehr vorkommt (Siehe Seite 4).

- Zuerst wird überprüft, ob es Auftragspositionen gibt, in denen das zu löschende Erzeugnis (fertigerznr = 200016) noch vorkommt.



Das zu löschende Erzeugnis kommt noch in einer Auftragsposition vor.

- Versucht man jetzt dieses Erzeugnis zu löschen, erhält man folgende Meldung:

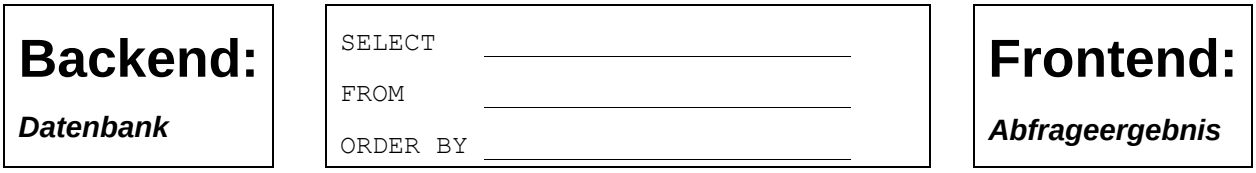


Ein Datensatz aus der Parent-Tabelle *fertigerzeugnisse* kann nicht gelöscht werden, solange sich noch ein Erzeugnis mit der zu löschenden *fertigerznr* (hier fertigerznr = 200016) in der Tabelle *auftragspositionen* befindet.

1.2SQL als Query Language (QL)

1.2.1 Einführung

Situation Für die Mitarbeiterinnen und Mitarbeiter der Vertriebsabteilung wird eine Übersicht (Liste) aller Kunden mit Firmenbezeichnung und E-Mail-Adresse gewünscht. Die Liste soll alphabetisch geordnet sein.



Abfragesprache SQL

Backend: In diesem Bereich liegen praktisch alle Dateien, die der MySQL-Server zur Verwaltung der Informationen braucht. Dieser Bereich ist den Administratoren, Programmierern und Designern vorbehalten.

Frontend: Dies ist der Bereich der Benutzer, also derjenigen, welche die Ergebnisse aus der Datenbank benötigen. Eine Software für ein Frontend ist beispielsweise die MySQL Workbench mit ihrer SQL-Development-Funktion.

Als Kommunikationsplattform zwischen *Backend* und *Frontend* dient die universelle Abfragesprache SQL **(Structured Query Language)**. Mit Hilfe der SELECT-Anweisung können Daten aus einer Datenbank abgefragt werden.

Befehlssyntax der SELECT-Anweisung

SELECT <attributbezeichnung1>, <attributbezeichnung2> … (* = alle Felder)
FROM <tabelle> [, <tabelle>, <tabelle>....]
[**WHERE** <bedingung>]
[**GROUP BY** <attributbezeichnung>]
[**HAVING** <bedingung>];
[**ORDER BY** <attributbezeichnung> [ASC|DESC]..]

Methodischer Hinweis:

Abfragen können in einer SQL-Datenbank gespeichert werden. Man spricht von Sichten (Views) auf eine Datenbank, die als virtuelle Tabellen verstanden werden können

Um die Abfragen in der SQL-Datenbank zu speichern ist die SQL-Anweisung

CREATE VIEW *viewname* AS SELECT ……;

anzuwenden.

Die so gespeicherten Abfragen können jederzeit mit dem SQL-Befehl

SELECT * FROM *viewname;*

erneut ausgeführt werden.

1.2.2 Auswahlabfragen am Beispiel der Tabelle *personal*

Mit Hilfe der Select-Anweisung werden Abfragen auf eine oder mehrere Tabellen durchgeführt. Die Select-Anweisung liefert wiederum eine Tabelle (Ergebnistabelle) zurück.

Die folgenden Aufgaben sind als einführende Beispiele in die SQL-Syntax zu verstehen. Grundlage der Aufgaben ist die Tabelle *personal* aus der Datenbank *muecke_1.*

Auszug aus der Datenbanktabelle *personal*:

personalnr	name	vorname	geschlecht	durchwahl	ortnr	strasse	gebtag	eintritt	steuerklasse	gehalt	kostnr
400000	Schneider	Heinrich	M	10	29306	Goezstraße 25	1966-10-18	1990-11-01	I	4800.00	02
400001	Schlauch	Franz	M	15	29675	Ulmer Weg 56	1962-04-01	1991-05-02	III	4400.00	05
400002	Schlauch	Karin	W	16	29675	Ulmer Weg 56	1966-09-10	1991-04-01	III	3675.00	04
400003	Böckle	Jennifer	W	21	29675	Hermann-Hesse-Str. 3	1966-05-25	1992-06-01	V	4600.00	03
400004	Hauffe	Johann	M	33	29306	Seestraße 21	1968-02-01	1991-03-02	III	5400.00	06
400005	Yilmaz	Ali	M	28	29315	Wehrstraße 87	1971-05-01	2000-07-01	III	2175.00	06
400006	Berger	Johann	M	41	30050	Vaihinger Str. 103	1956-10-18	2001-05-01	I	1900.00	05
400007	Schweizer	Anton	M	22	29303	Schillingstraße 18	1962-01-30	2001-05-01	III	4950.00	05
400008	Lambert	Hans	M	26	29306	Molkereigässle 7	1968-05-29	2003-04-02	III	3450.00	06
400009	Sautter	Fritz	M	29	28952	Karlstraße 23	1984-08-12	2004-10-01	V	1900.00	06
400010	Branduardi	Francesco	M	30	29740	Rosenweg 11A	1979-10-31	2005-01-02	I	6150.00	07

Aufgaben

Ausgabe einer ausgewählten Anzahl von Attributen

Abfrage 1: Alle Mitarbeiter sollen mit Vorname, Nachname, Durchwahlnummer und Kostenstellennummer aufgelistet werden. Die Liste soll nach den Kostenstellennummern absteigend geordnet werden. Die Mitarbeiter der einzelnen Kostenstellen sind in alphabetischer Reihenfolge aufzulisten.

Selektion mit einer Bedingung

Abfrage 2: Alle Mitarbeiter mit dem Nachnamen ‘Schlauch‘ sollen mit Vorname, Name und Gehalt aufgelistet werden.

Abfrage 3: Alle Mitarbeiter in deren Nachname der Name ‘Schlauch‘ enthalten ist, sollen mit Vorname, Name und Gehalt aufgelistet werden.

Abfrage 4: Alle Mitarbeiter, deren Gehalt über 5000 Euro liegt, sollen mit Vorname, Name, Geburtsdatum und Gehalt aufgelistet werden.

Abfrage 5: Von allen Mitarbeitern, die vor dem 01.01.1970 geboren sind, sollen Name, Geburtstag und Tag des Eintritts in das Unternehmen aufgelistet werden.

Abfrage 6: Für alle Mitarbeiter der Kostenstelle 04 soll das Jahresgehalt ermittelt werden. Jeder Mitarbeiter dieser Kostenstelle erhält 13 Monatsgehälter.
(Attribute: Vorname, Name, Kostenstellennummer, Jahresgehalt)

Selektion mit mehreren Bedingungen

Abfrage 7: Aufzulisten sind alle Mitarbeiter**innen** (Geschlecht: W) mit der Steuerklasse II. (Attribute: Vorname, Name , Steuerklasse)

Abfrage 8: Aufzulisten sind alle Mitarbeiter, die in den Abteilungen mit den Kostenstellennummern 02 und 05 beschäftigt sind.
(Attribute: Vorname, Name , Kostenstellennummer)

Selektion mit verneinender Bedingung

Abfrage 9: Alle Mitarbeiter, die nicht in der Abteilung mit der Kostenstellennummer 06 beschäftigt sind, sollen mit Vorname, Name und Kostenstellennummer aufgelistet werden. Die Auflistung soll nach der Kostenstellennummer absteigend sortiert werden.

Selektion mit Wertebereichsbedingung

Abfrage 10: Alle Mitarbeiter, deren Gehalt zwischen 1.800,00 und 2.000,00 Euro liegt (jeweils einschließlich), sollen mit Name und Gehalt aufgelistet werden. Die Auflistung soll nach dem Gehalt aufsteigend geordnet sein.

Selektion mit Datumsfunktionen

Abfrage 11: Alle Mitarbeiter, die seit dem Jahr 2010 im Unternehmen beschäftigt sind, sollen mit Vorname, Name und Eintrittstermin aufgelistet werden.

Abfrage 12: Aufzulisten sind alle Mitarbeiter, die im Monat Mai Geburtstag haben (alle Attribute). Die Auflistung ist so zu ordnen, dass die jüngsten Mitarbeiter an erster Stelle stehen.

Abfrage 13: Es ist eine Liste derjenigen Mitarbeiter zu erstellen, die im jeweilig aktuellen Jahr ihr 10jähriges Betriebsjubiläum haben.

Selektion mit Aggregat-Funktionen

Abfrage 14: Von allen Mitarbeitern sind die gesamten Monatsgehälter, das durchschnittliche sowie das höchste und das niedrigste Gehalt aufzulisten (siehe Abb.).

Gehaltssumme	durchschnittliches Gehalt	hoechstes Gehalt	geringtes_Gehalt
222537.50	2853.044872	6150.00	1800.00

Abfrage 15: Mit Hilfe einer Abfrage ist die Anzahl der Mitarbeiter, die in der Abteilung mit der Kostenstellennummer 04 beschäftigt sind, zu ermitteln.

kostnr	Anzahl_Mitarbeiter
04	12

1.2.3 Lösungshinweise zu Auswahlabfragen am Beispiel der Tabelle *personal*

Abfrage 1

```
SELECT vorname, name, durchwahl, kostnr
FROM personal
ORDER BY kostnr DESC, name, vorname;
```

Anmerkung:
- Jede SELECT-Anweisung beginnt mit einer Projektion.
  Die **Projektion** von Daten aus einer Tabelle ermöglicht es, nur bestimmte Attribute (Spalten) aus der Tabelle auszuwählen. Die Projektion von Daten entspricht somit der vertikalen Auswahl bestimmter Datenfelder.
  Das *-Zeichen wählt alle Attribute einer Tabelle aus (vollständige Projektion).
- Die FROM-Klausel bestimmt die Tabelle/(n), in der/(denen) die gewünschten Daten gespeichert sind.
- Mit Hilfe der ORDER BY – Klausel kann eine Ergebnisliste sortiert werden. Vorgabe: ASC für aufsteigend; sonst Zusatz DESC für absteigend.

Abfrage 2

```
SELECT vorname, name, gehalt
FROM personal
WHERE name = "Schlauch";
```

Anmerkung:
- Die Auswahl bestimmter Zeilen einer Datenbanktabelle wird als Selektion bezeichnet. Die Selektion von Daten entspricht somit der horizontalen Auswahl bestimmter Datensätze.
- Die Auswahl erfolgt mit Hilfe von Bedingungen innerhalb der WHERE-Klausel.
- Aufbau einer Bedingung:
  attributbezeichnung vergleichsoperator vergleichswert
  z.B. name = “Huber“
- Bei einer Bedingung mit einfachem Textvergleich steht der Vergleichswert in Anführungszeichen.

Abfrage 3

```
SELECT vorname, name, gehalt
FROM personal
WHERE name LIKE "%Schlauch%";
```

Anmerkung:
- Soll eine Selektion nicht durch einen präzisen Vergleichswert sondern nur durch Teile davon erfolgen, muss im Vergleichswert ein Platzhalter eingefügt werden. In diesem Falle lautet der Vergleichsoperator LIKE.
- MySQL verwendet als Platzhalter für beliebige Zeichen innerhalb einer Zeichenkette das %-Symbol.
- Platzhalter für **ein** beliebiges Zeichen ist das _-Symbol (Underline).
  z.B. name LIKE “Ma_er“

Abfrage 4

```
SELECT vorname, name, gebtag, gehalt
FROM personal
WHERE gehalt > 5000;
```

Anmerkung: ➢ Bei einer Bedingung mit numerischem Vergleich steht der Vergleichswert ohne Anführungszeichen.
z.B. gehalt < 1000

Abfrage 5

```
SELECT name, gebtag, eintritt
FROM personal
WHERE gebtag < "1970-01-01";
```

Anmerkung: ➢ Bei einer Bedingung, die einen Vergleich mit einem Datumswert enthält, erfolgt die Datumswertangabe in folgender Schreibweise: „yyyy-mm-tt“.
z.B. eintritt > “2010-10-31“

Abfrage 6

```
SELECT vorname, name, kostnr, gehalt * 13 AS Jahresgehalt
FROM personal
WHERE kostnr ="04";
```

Anmerkung: ➢ Ergebnisse von Rechenoperationen (berechnete Felder) können in die Projektion von Daten aufgenommen werden. Das Rechenergebnis wird mit ‘AS‘ einer Variablen zugewiesen.
z.B. SELECT name, gehalt / 180 AS Stundenlohn …

Abfrage 7

```
SELECT vorname, name, steuerklasse
FROM personal
WHERE geschlecht = "W"
AND steuerklasse ="II";
```

Anmerkung: ➢ Die Auswahl von Datenbankinhalten kann durch mehrere Bedingungen eingeschränkt werden.
Sollen alle Selektionsbedingungen erfüllt sein, so müssen die einzelnen Bedingungen mit dem Operator AND verknüpft werden.
(Sowohl Bedingung 1 als auch Bedingung 2 …. **müssen** erfüllt sein!)
z.B. gehalt > 5000 AND kostnr = “03“
(Siehe hierzu auch „1.4.5 SQL und Boolesche Algebra“ – Seite 118ff).

Abfrage 8

```
SELECT vorname, name, kostnr
FROM personal
WHERE kostnr = "02"
OR kostnr ="05";
```

Anmerkung:
- Soll mindestens eine der Selektionsbedingungen erfüllt sein, so müssen die einzelnen Bedingungen mit dem Operator OR verknüpft werden. (Entweder Bedingung 1, oder Bedingung 2 …. oder alle Bedingungen **können** erfüllt sein!)
  z.B. ortnr = 29521 OR ortnr = 29531

Abfrage 9

```
SELECT vorname, name, kostnr
FROM personal
WHERE NOT kostnr = "06"                    (alternativ: kostnr != "06")
ORDER BY kostnr DESC;
```

Anmerkung:
- Soll eine Selektionsbedingung nicht erfüllt sein, so muss der Bedingung der Operator NOT vorangestellt werden.
  z.B. NOT steuerklasse = "III"
- Alternativ kann eine solche ausgrenzende Bedingung auch mit Hilfe des Operators != (ungleich) formuliert werden.
  z.B. steuerklasse != "III"

Abfrage 10

```
SELECT name, gehalt
FROM personal
WHERE gehalt BETWEEN 1800 AND 2000
ORDER BY gehalt;
```

Anmerkung:
- Soll die Auswahl von Datenbankinhalten in Abhängigkeit von Werten innerhalb eines Wertebereichs erfolgen, muss folgende Syntax angewandt werden:
  attributbezeichnung BETWEEN *untergrenze* AND *obergrenze*
  z.B. gehalt BETWEEN 2000 AND 3000
- Die ausgewählten Daten schließen die Ober- und Untergrenze mit ein.

Abfrage 11

```
SELECT vorname, name, eintritt
FROM personal
WHERE YEAR(eintritt) = 2010;
```

Anmerkung:
- Die Funktion YEAR(*datumfeld*) gibt die Jahreszahl aus einem regulären Datum (Datentyp Zahl) zurück.

Abfrage 12

```
SELECT *
FROM personal
WHERE MONTH(gebtag) = 5
ORDER BY gebtag DESC;
```

Anmerkung:
- Die Funktion MONTH(*datumfeld*) gibt die Monatszahl aus einem regulären Datum (Datentyp Zahl) zurück.

Abfrage 13

```
SELECT *
FROM personal
WHERE YEAR(NOW()) - YEAR(eintritt) = 10;
```

Anmerkung:
- Die Funktion NOW() gibt das aktuelle Systemdatum mit Uhrzeit (Datentyp DATE) zurück.

Abfrage 14

```
SELECT SUM(gehalt) AS Gehaltssumme,
    AVG(gehalt) AS "durchschnittliches Gehalt" ,
    MAX(gehalt) AS "hoechstes Gehalt",
    MIN(gehalt) AS geringtes_Gehalt
FROM personal;
```

Anmerkung:
- Aggregat-Funktionen führen auf Basis einer Wertemenge Berechnungen durch und geben deren Ergebnis zurück.
  (Die folgenden Funktionen können nur bei numerischen Attributen verwendet werden.)
  - Die Funktion SUM(attributbezeichnung) gibt die Summe aller Werte des Attributs zurück.
  - Die Funktion AVG(attributbezeichnung) gibt den Mittelwert der Werte des Attributs zurück (numerische Attribute).
  - Die Funktion MAX(attributbezeichnung) gibt den höchsten Wert des Attributs zurück (numerische Attribute).
  - Die Funktion MIN(attributbezeichnung) gibt den niedrigsten Wert des Attributs zurück (numerische Attribute).

Abfrage 15

```
SELECT kostnr, COUNT(*) AS Anzahl_Mitarbeiter
FROM personal
WHERE kostnr = "04";
```

Anmerkung:
- Die Funktion COUNT(*) ermittelt die Anzahl von Datensätzen einer Tabelle. Zählt auch Duplikate und NULL-Werte.

1.2.4 Übersicht der Hauptklauseln der Select-Anweisung

Die **FROM** - Klausel	Hinter **FROM** steht der Name der Tabelle. Entstammen die Felder verschiedenen Tabellen, so muss jeweils der Tabellenname durch einen Punkt getrennt vorangesetzt werden.
Die **WHERE** - Klausel	Damit wird bestimmt, welche Datensätze (Zeilen einer Tabelle) ausgewählt werden sollen. (**Bedingungsprüfung**).
Die **ORDER BY** - Klausel	Daten werden nach einem oder mehreren Feldnamen **sortiert** ausgegeben. Die **vorgegebene Sortierreihenfolge ist *aufsteigend ASC***ending). Soll absteigend sortiert werden, muss (***DESC***ending) eingegeben werden.
Die **GROUP BY** - Klausel	Sie dient dazu, die Zeilen einer Tabelle nach bestimmten Feldern zu gruppieren.
Die **HAVING** - Klausel	Bei der **HAVING** - Klausel werden bestimmte Gruppen aus einer Gruppenmenge ausgewählt. Somit kann die **HAVING** - Klausel nur im Zusammenhang mit der **GROUP BY** - Klausel angewandt werden.

Übersicht: Vergleichs-Operatoren / Arithmetische Operatoren

SQL kennt die üblichen Vergleichsoperatoren:
= gleich != *bzw.* <> ungleich > größer < kleiner >= größer gleich <= kleiner gleich

BETWEEN *.Wert1..***AND***..Wert2..*	Vergleichswert liegt *zwischen* Wert1 und *Wert2*
IN *Werteliste*	Vergleichswert ist in der angegebenen *Werteliste*
Like *Zeichenfolge*	Vergleichszeichen entsprechen der *Zeichenfolge*
Is Null *Feld*	Vergleichsfeld hat einen NULL-Wert
Vergleichsoperatoren können verknüpft werden mit den Operatoren AND, OR und NOT	

Rechenoperationen (+ , - , * , /) sind nur möglich mit numerischen Daten.

Übersicht Aggregatsfunktionen (Gruppierungsfunktionen)

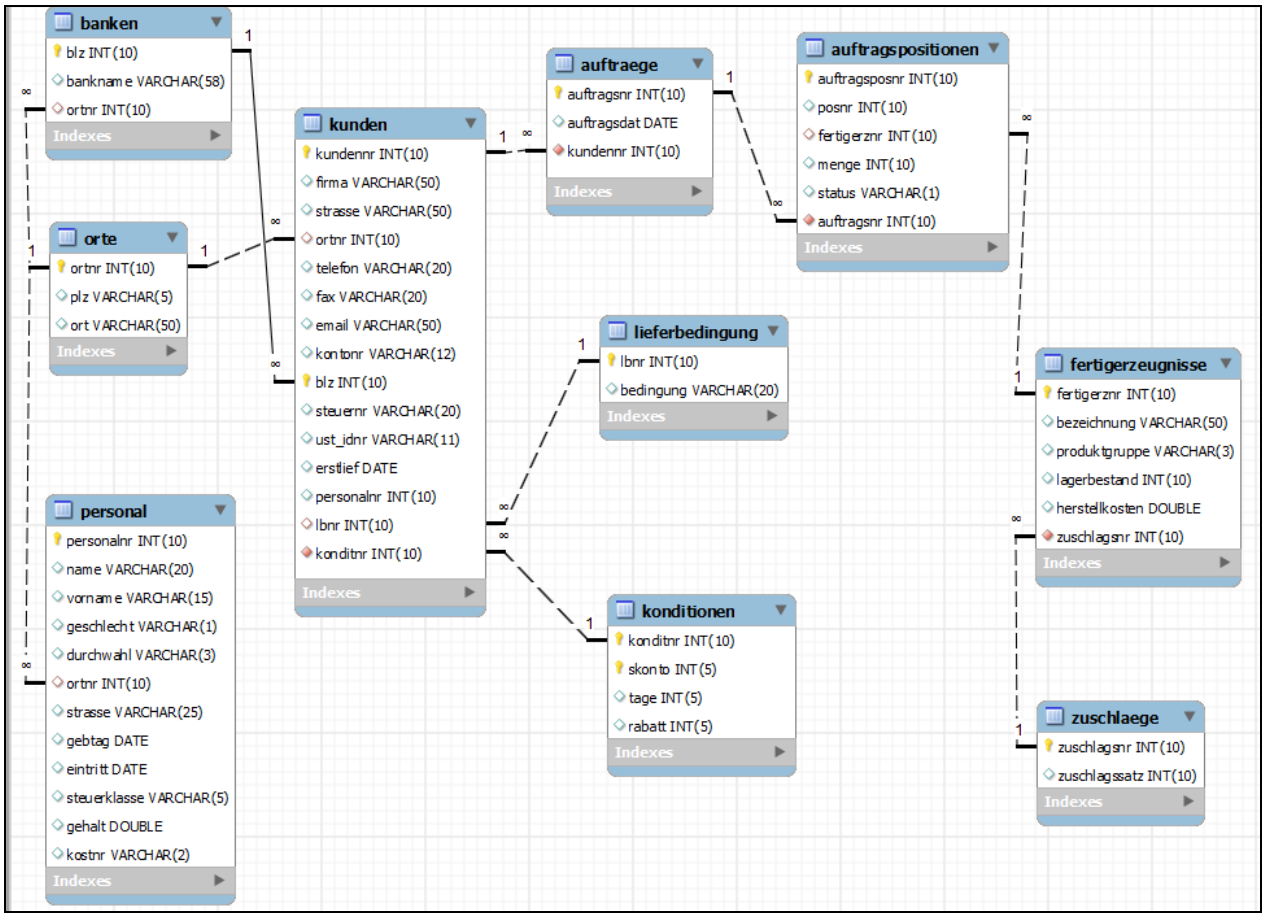
AVG(Spalte)	= Durchschnittswert	**COUNT**(Spalte)	= Anzahl aller Einträge
MAX(Spalte)	= Maximalwert	**MIN**(Spalte)	= Minimalwert
SUM(Spalte)	= Summe aller Einträge in einer Spalte		

Gruppierungsfunktionen können nur anstelle eines Spaltennamens direkt hinter der SELECT-Anweisung stehen. Sie liefern genau einen Wert, beziehen sich jedoch auf mehrere Tabellenzeilen.

1.2.5 Auswahlabfragen mit mehreren Tabellen

Als Datenbasis zur Lösung der nachfolgenden Aufgabenstellungen steht die Datenbank ***muecke_2*** zur Verfügung. Sie enthält weitere Tabellen. Darüber hinaus enthalten einige Tabellen geänderte Daten. Mit Hilfe der zum Download bereitgestellten Skriptdateien lassen sich die Tabellenstruktur (*muecke2_struktur.sql*) und die Tabelleninhalte (*muecke2_daten.sql*) importieren.

EERM der Datenbank *muecke_2*



Situation:

Für eine Werbeaktion wird eine Adressliste aller Kunden aus Stuttgart und Ulm benötigt. Die Liste die Attribute n*ame, plz, strasse* und *ort* enthalten und nach den Kundennamen alphabetisch sortiert sein.

kundennr	firma	strasse	ortnr	ortnr	ort
270033	Einrichtungshaus Blessing	Bissinger Str. 5	34976	34976	Augsburg, Bay
270036	Meinschneider Einrichtungen	Brunnenweg 88	29303	29303	Esslingen am Neckar
270017	NimmMit Mitnahmemöbel GmbH	Brühlstr. 50	29306	29306	Esslingen am Neckar
270053	Einrichtungshaus Bluthardt	Kastellstr. 14	29303	29303	Esslingen am Neckar
270040	Schober Vollholzmöbel KG	Heilbronner Str. 3	29675	29675	Stuttgart
270002	FIRENCE Möbel & Leuchten GmbH	Königsstraße 12	29521	29521	Stuttgart
270018	Einrichtungshaus Bull e.K.	Hauptstraße	29654	29654	Stuttgart
270008	Bär-Möbel GmbH	Luisenstr. 55	36023	36023	Ulm, Donau

Daten aus der Tabelle *kunden*

Daten aus der Tabelle *orte*

Ein Datensatz (Zeile) wird ausgewählt, wenn
- die Attributwerte des gemeinsamen Attributs ortnr übereinstimmen,
- und die Auswahlbedingung (ort = Stuttgart oder ort = Ulm) erfüllt ist.

```
SELECT kundennr, firma, strasse, kunden, plz, ort      ← (1)
FROM kunden, orte                                      ← (2)
WHERE kunden.ortnr = orte.ortnr                        ← (3)
AND (ort LIKE "Ulm%" OR ort LIKE "Stutt%")             ← (4)
ORDER BY ort;                                          ← (5)
```

(1) Zusammenstellen der anzuzeigenden Attribute (Projektion).

(2) Auswahl der benötigten Tabellen.

(3) Bedingung für die Verbindung der beiden Tabellen. Für das gemeinsame Attribut ortnr müssen in beiden Tabellen übereinstimmende Attributswerte vorhanden sein. Attribute, die in beiden Tabellen vorkommen, müssen mit dem Tabellennamen als „Vorsilbe“ angesprochen werden, Beispiel *kunden.ortnr.*

(4) Bedingung für die Auswahl der Datensätze (= Selektion).
(Die in der WHERE-Klausel verwendeten Platzhalter sollen sicher stellen, dass auch eine Ortsbezeichnung wie beispielsweise „Ulm, Donau“ selektiert wird.)

(5) Sortierung.

Übungsaufgaben zu Auswahlabfragen mit mehreren Tabellen

Abfrage 1 Alle Kunden mit einem Rabattsatz von 10% sollen mit den Feldern *firma, plz, ort* und *rabatt* aufgelistet werden.

Abfrage 2 Für welche Kunden gilt die Lieferbedingung „ab Werk“ (Attribute *kundennr, firma, plz, strasse, ort*)

Abfrage 3 Die Vertriebsleiterin Frau Angelika Schlauch-Köpf wünscht von der Produktgruppe I der Fertigerzeugnisse eine Auflistung der Attribute *erzeugnisnr, bezeichnung, herstellkosten* sowie Zuschlagssatznummer (Attribut *zuschlagsnr*) und Kalkulationszuschlagssatz (Attribut *zuschlagssatz*)

Abfrage 4 Die Verkaufsleitung wünscht eine Aufstellung der Herstellkosten, Zuschlagssätze und der Netto-Verkaufspreise aller Fertigerzeugnisse.
(Netto-Verkaufspreis = Herstellkosten * Zuschlagssatz / 100)
Die Aufstellung soll nach den Verkaufspreisen aufsteigend sortiert sein und folgendes Aussehen haben (auszugsweise):

fertigerznr	bezeichnung	herstellkosten	zuschlagssatz	Verkaufspreis
200028	Tisch ALPHA	98.50	16	114.260000
200023	Wandregal BETA	175.00	16	203.000000
200034	Rollcontainer UNIT	175.50	20	210.600000

Abfrage 5 Welche Fertigerzeugnisse hat der Kunde City-Möbel GmbH (kundennr = 270042) im Juni 2013 in Auftrag gegeben?
Die Aufstellung soll folgendes Aussehen haben (auszugsweise):

kundennr	firma	plz	ort	auftragsdat	fertigerznr	menge	bezeichnung
270042	City-Möbel GmbH	73312	Geislingen an der Steige	2013-06-17	200006	5	Schrankwand OSTSEE
270042	City-Möbel GmbH	73312	Geislingen an der Steige	2013-06-17	200011	4	Sideboard GEMINI
270042	City-Möbel GmbH	73312	Geislingen an der Steige	2013-06-07	200014	2	Sideboard MENOS
270042	City-Möbel GmbH	73312	Geislingen an der Steige	2013-06-22	200014	7	Sideboard MENOS

Abfrage 6 Die Auftragskopfdaten des Auftrages mit der Auftragsnummer 20370 sind wie in der nachfolgenden Anzeige dargestellt, anzuzeigen.

auftragsnr	auftragsdat	kundennr	firma	strasse	plz	ort
20370	2014-08-20	270015	Schlenz Möbel KG	Bachrain 71	89250	Senden, Iller

Abfrage 7 Die Positionen des Auftrages mit der Auftragsnummer 20370 sind auszuwählen und wie nachfolgend dargestellt, anzuzeigen:

auftragsnr	auftragsposnr	fertigerznr	bezeichnung	menge	Verkaufspreis	Gesamt	rabatt	Zielverkaufspreis
20370	1629	200030	Stuhl EIFEL	12	319.580000	3834.960000	10	3451.4640000000
20370	1630	200005	Schrankwand AZZURO	7	900.000000	6300.000000	10	5670.0000000000

Abfrage 8 Für den Auftrag mit der Auftragsnummer 20370 (siehe Abfrage 7) sind die Zielverkaufspreise der Auftragspositionen zusammenzuzählen und als Gesamtzielverkaufspreis anzuzeigen:

Gesamtzielverkaufspreis
9121.4640000000

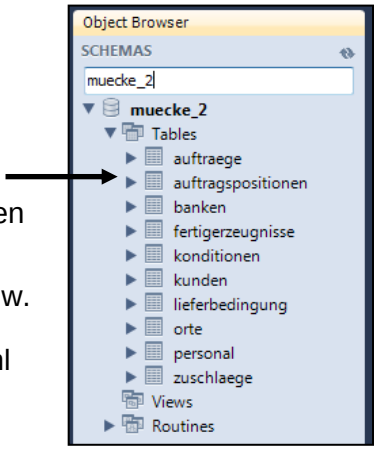
1.2.6 Datenbankview

Mit der SQL-Anweisung

```
CREATE VIEW viewname AS SELECT <attributbezeichnung1>,
<attributbezeichnung2>…
FROM <tabelle> [, <tabelle>, <tabelle>....]
[WHERE <bedingung(en)> ]
```

- können Abfragen als eigenständige Objekte der Datenbank gespeichert werden. Es werden die Abfragen und nicht die Ergebnisse gespeichert.
- *Views* können in weiteren Abfragen wie jede andere Tabelle benutzt werden.
- Durch den Einsatz von Datenbankviews können längere Abfragen mit Rechenfeldern vereinfacht werden.

  Hinweis: **`DROP VIEW viewname`** löscht einen angelegten View.



Die so gespeicherten Abfragen können jederzeit mit dem SQL-Befehl

```
Select * FROM viewname;
```

erneut ausgeführt werden.

Hinweis für den Unterricht:
Datenbankviews sind nicht Bestandteil des Lehrplans in den Fächern Wirtschaftsinformatik für das Wirtschaftsgymnasium und Informatik für das Berufliche Gymnasium. Gleichwohl ist ihr Einsatz im Unterricht als eine Möglichkeit denkbar, komplexere Aufgabenstellungen mithilfe von Views zu strukturieren und zu vereinfachen.

Beispiel: *Für den Artikel „Kleiderschrank Paris“ (fertigerznr 200001) sollen der Nettoverkaufspreis und der Bruttoverkaufspreis (zuzüglich 19 % Umsatzsteuer) ausgewiesen werden.*

Dazu wird zunächst

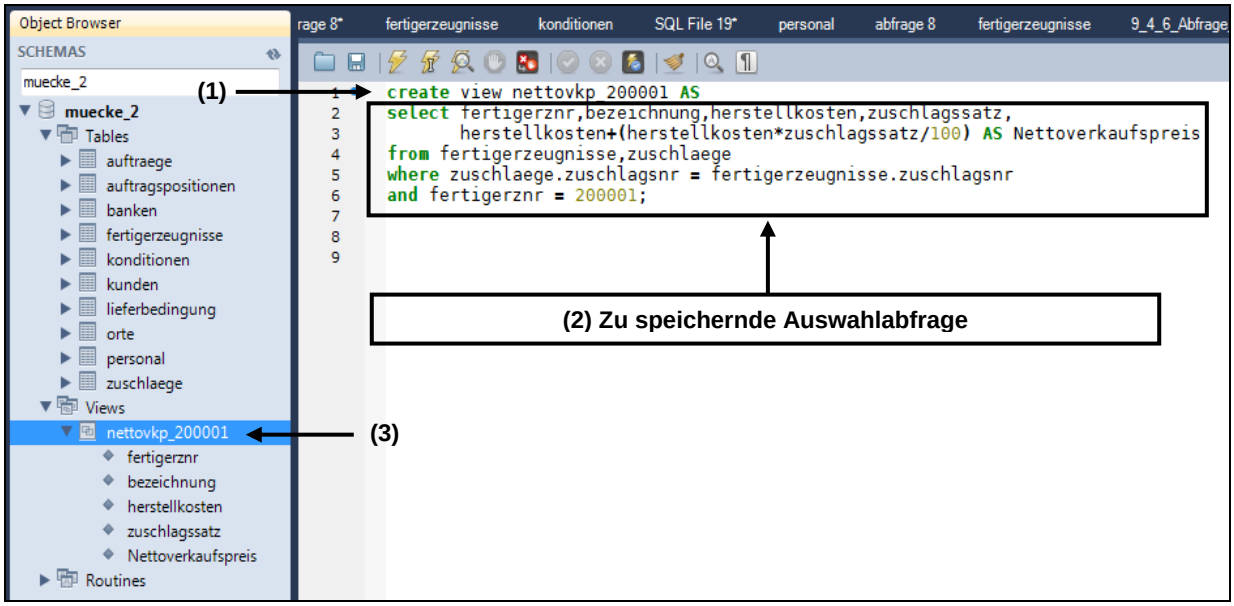
die Abfrage zum Ermitteln des Nettoverkaufspreises gebildet:

```
SELECT fertigerznr, bezeichnung, herstellkosten, zuschlagssatz, herstellkosten +
                  (herstellkosten * zuschlagssatz / 100) AS Nettoverkaufspreis
FROM fertigerzeugnisse, zuschlaege
WHERE zuschlaege.zuschlagsnr = fertigerzeugnisse.zuschlagsnr
AND fertigerznr = 200001;
```

die Abfrage liefert das folgende Ergebnis:

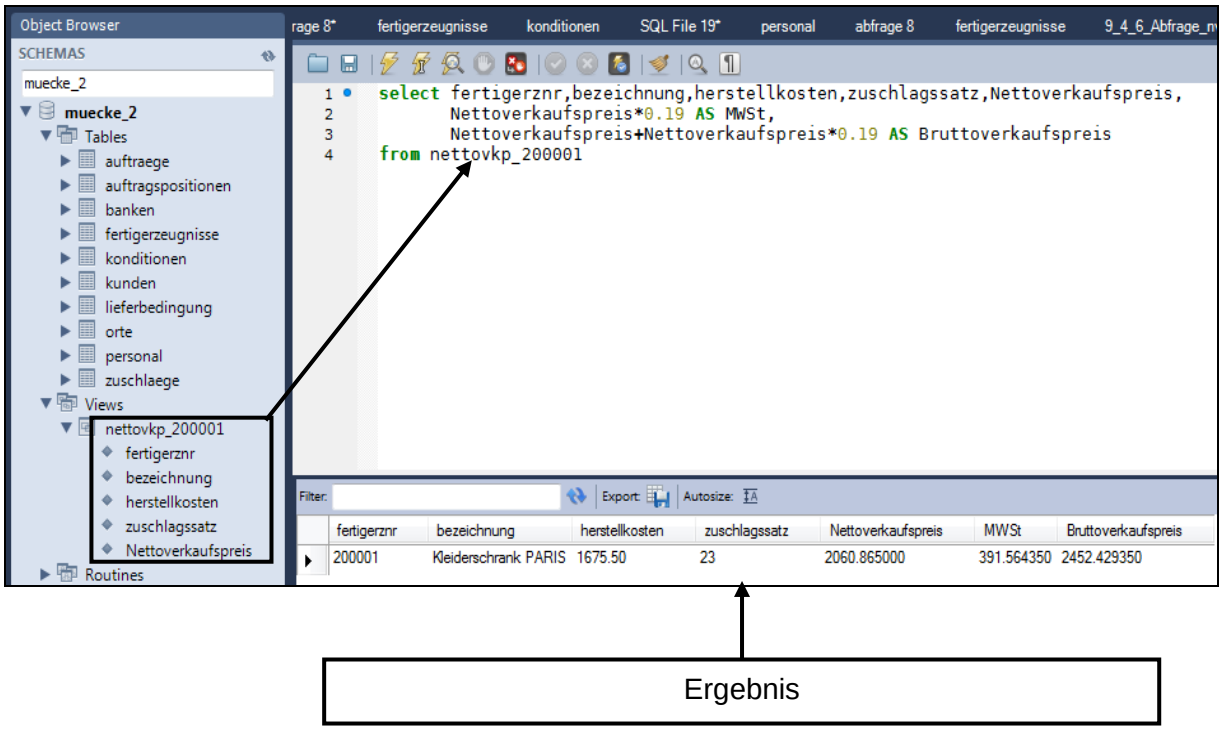
fertigerznr	bezeichnung	herstellkosten	zuschlagssatz	Nettoverkaufspreis
200001	Kleiderschrank PARIS	1675.50	23	2060.865000

Jetzt wird die Abfrage (2) als View unter dem Namen „nettovkp_200001“ gespeichert (1)



Der View erscheint als eigenständiges Objekt im Object Browser (3).

(2) Nun kann der View als eigenständige Tabelle zum Berechnen der Umsatzsteuer und des Bruttoverkaufspreises herangezogen werden:



Weitere Übungsbeispiele

Abfrage 9 Da die Abfrage 7 (Auftragspositionen mit den jeweiligen Gesamtwerten) als Grundlage für die Abfrage 8 dient, kann die Abfrage 7 auch in der Datenbank als Datenbankview gespeichert werden.

Vorgehensweise

a) Erstellen eines Datenbankview mit Namen *apos_20370* in dem die Select-Anweisung der Abfrage 7 zum Anzeigen der Auftragspositionen des Auftrages abgespeichert wird.

b) Ermitteln der Gesamtsumme (=Gesamtzielverkaufspreis) für die Auftragspositionen des Auftrages mit der Auftragsnummer 20370. Dazu wird der Datenbankview *apos_20370* verwendet.

Abfrage 10 Zum Gesamtzielverkaufspreis (siehe Abfrage 9) sind die zur Zeit aktuelle Umsatzsteuer und der Bruttoverkaufspreis zu ermitteln und anzuzeigen.

gewünschtes Ergebnis:

Gesamtzielverkaufspreis	MWSt	Bruttoverkaufspreis
9121.4640000000	1733.0781600000	10854.5421600000

Hinweis: Die Abfrage 9 ist als View zu speichern. Mit Hilfe des Views können dann die Umsatzsteuer und der Bruttoverkaufspreis bequemer ermittelt und angezeigt werden.

Abfrage 11 Aus dem Bruttoverkaufspreis sollen dann mit Hilfe der Konditionen für den Kunden mit der Kundennummer 270015 (siehe Abfrage 6) der Skontobetrag und der Barpreis ermittelt und angezeigt werden.

Die Abfrage zum Ermitteln des Bruttoverkaufspreises (siehe Abfrage 10) wird jetzt als View gespeichert. Mit Hilfe des Views können dann der Skontobetrag und der Barpreis ermittelt werden.

gewünschtes Ergebnis:

Bruttoverkaufspreis	skonto	Skonto	Barpreis
10854.54216	3	325.63626480000005	10528.905895200001

Lösungsvorschläge

Abfrage 1

```
SELECT firma, plz, ort, rabatt
FROM kunden, konditionen, orte
WHERE kunden.ortnr = orte.ortnr
AND kunden.konditnr = konditionen.konditnr
AND rabatt = 10;
```

Abfrage 2

```
SELECT kundennr, firma, plz, strasse, ort
FROM kunden, orte, lieferbedingung
WHERE lieferbedingung.lbnr = kunden.lbnr
AND kunden.ortnr = orte.ortnr
```

```
AND bedingung = "ab Werk";
```

Abfrage 3

```
SELECT fertigerznr, bezeichnung, herstellkosten,
                        fertigerzeugnisse.zuschlagsnr, zuschlagssatz
FROM fertigerzeugnisse, zuschlaege
WHERE fertigerzeugnisse.zuschlagsnr = zuschlaege.zuschlagsnr
AND produktgruppe = "I";
```

Anmerkung:

Ist ein Datenfeld in verschiedenen Tabellen erfasst (Primär-/Fremdschlüssel), so muss der Tabellenname durch einen Punkt getrennt vorangesetzt werden.

Abfrage 4

```
SELECT fertigerznr,bezeichnung, herstellkosten, zuschlagssatz,
   herstellkosten+(herstellkosten*zuschlagssatz/100) AS Verkaufspreis
FROM zuschlaege, fertigerzeugnisse
WHERE fertigerzeugnisse.zuschlagsnr = zuschlaege.zuschlagsnr
ORDER BY verkaufspreis;
```

Anmerkung:

Der Verkaufspreis wird mit Hilfe eines Rechenfeldes ermittelt.

Sollen die Spaltenüberschriften von den Attributbezeichnungen abweichen (z. B. einheitliche Großschreibweise), kann dies mit Alias-Namen erreicht werden. Werden im Alias-Namen Sonderzeichen oder Leerzeichen verwendet, muss er zwischen Anführungszeichen stehen.

Beispiel: Abfrage 4 mit Alias-Namen

```
SELECT fertigerznr AS "Fert.-Erz.-Nr", bezeichnung AS Bezeichnung,
   herstellkosten AS Herstellkosten, zuschlagssatz AS Zuschlagssatz,
   herstellkosten+(herstellkosten*zuschlagssatz/100) AS Verkaufspreis
FROM zuschlaege, fertigerzeugnisse
WHERE fertigerzeugnisse. Zuschlagsnr = zuschlaege.zuschlagsnr
ORDER BY verkaufspreis;
```

erzeugt folgende Ausgabe:

Fert.-Erz.-Nr	Bezeichnung	Herstellkosten	Zuschlagssatz	Verkaufspreis
200028	Tisch ALPHA	98.5	16	114.26
200023	Wandregal BETA	175	16	203

Abfrage 5

```
SELECT kunden.kundennr, firma, plz, ort, auftragsdat,
             auftragspositionen.fertigerznr, menge, bezeichnung
FROM kunden, orte, auftraege, auftragspositionen, fertigerzeugnisse
WHERE kunden.ortnr = orte.ortnr
AND kunden.kundennr = auftraege.kundennr
AND auftraege.auftragsnr = auftragspositionen.auftragsnr
AND auftragspositionen.fertigerznr= fertigerzeugnisse.fertigerznr
AND YEAR(auftragsdat)=2013
AND MONTH(auftragsdat)=  6
AND firma LIKE "City-Möbel%";
```

Anmerkung:

- Für die Datenauswahl sind fünf Tabellen zu verbinden.
- Die Funktion YEAR(Datumswert) liefert aus einem Datumswert die Jahresangabe als Integer-Wert
- Die Funktion MONTH(Datumswert) liefert aus einem Datumswert die Monatsangabe als Integer-Wert.

Abfrage 6

```
SELECT auftragsnr, auftragsdat, auftraege.kundennr, firma,
                                                strasse, plz, ort
FROM auftraege, kunden, orte
WHERE auftraege.kundennr = kunden.kundennr
AND kunden.ortnr = orte.ortnr
AND auftragsnr = 20370;
```

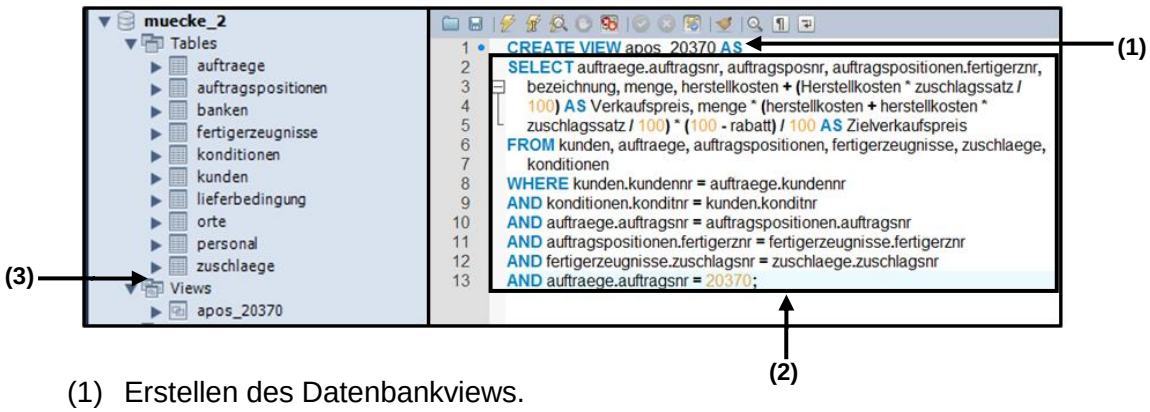
Abfrage 7

```
SELECT auftraege.auftragsnr, auftragsposnr, auftragspositionen.fertigerznr,
    bezeichnung, menge, herstellkosten+(herstellkosten*zuschlagssatz/100)
    AS Verkaufspreis , menge * (herstellkosten + herstellkosten *
    Zuschlagssatz / 100) AS Gesamt, rabatt, menge * (herstellkosten +
    herstellkosten * zuschlagssatz / 100) * (100 - rabatt) / 100 AS
    Zielverkaufspreis
FROM kunden, auftraege, auftragspositionen, fertigerzeugnisse, zuschlaege,
    konditionen
WHERE kunden.kundennr = auftraege.kundennr
AND konditionen.konditnr = kunden.konditnr
AND auftraege.auftragsnr = auftragspositionen.auftragsnr
AND auftragspositionen.fertigerznr = fertigerzeugnisse.fertigerznr
AND fertigerzeugnisse.zuschlagsnr = zuschlaege.zuschlagsnr
AND auftraege.auftragsnr = 20370;
```

Abfrage 8

```
SELECT SUM(menge * (herstellkosten + herstellkosten * zuschlagssatz /
    100) * (100 - rabatt) / 100) AS Gesamtzielverkaufspreis
FROM kunden, auftraege, auftragspositionen, fertigerzeugnisse,
    zuschlaege, konditionen
WHERE kunden.kundennr = auftraege.kundennr
AND konditionen.konditnr = kunden.konditnr
AND auftraege.auftragsnr = auftragspositionen.auftragsnr
AND auftragspositionen.fertigerznr = fertigerzeugnisse.fertigerznr
AND fertigerzeugnisse.zuschlagsnr = zuschlaege.zuschlagsnr
AND auftraege.auftragsnr = 20370;
```

Abfrage 9a Erstellen des Datenbankviews mit dem Namen *apos_20370.* Grundlage für den View ist die Abfrage 7.



(1) Erstellen des Datenbankviews.

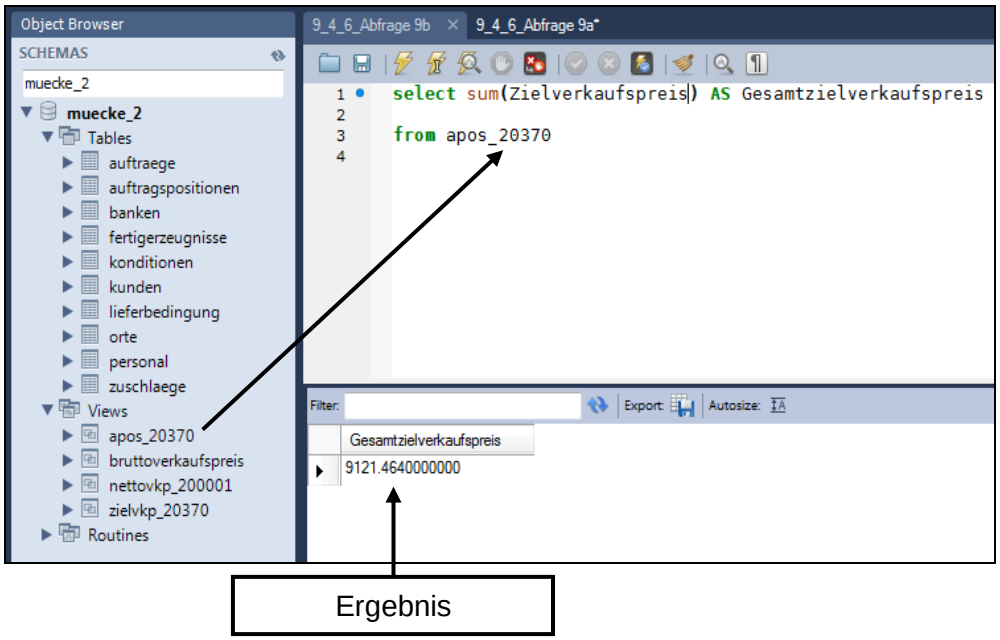
(2) Select-Anweisung, die gespeichert wird.

(3) Der Datenbankview wird in der Datenbank gespeichert.

Abfrage 9b Aufrufen des Datenbankviews

Für die gespeicherten Auftragspositionen des Auftrages mit der Auftragsnummer 20370 sind jetzt die Gesamtwerte zusammenzuzählen und die

Auftragssumme anzuzeigen.
Dazu wird der eingerichtete Datenbankview in der FROM-Klausel verwendet.



Abfrage 10a Erstellen des Datenbank-Views mit dem Namen *zielvkp_20370*

```
CREATE VIEW zielvkp_20370 AS
    SELECT SUM(Zielverkaufspreis) AS Gesamtzielverkaufspreis
    FROM apos_20370;
```

Erläuterung: Die SQL-Anweisung zum Aufsummieren der Zielverkaufspreise der einzelnen Auftragspositionen (Abfrage 9a) wird im Datenbankview *zielvkp_20370*, der auf den View *apos_20370* aufbaut, gespeichert.

Abfrage 10b Der Gesamtzielverkaufspreis (Datenbankview *zielvkp_201370*) wird verwendet, um die Umsatzsteuer und den Bruttoverkaufspreis zu berechnen.

```
SELECT Gesamtzielverkaufspreis, Gesamtzielverkaufspreis * 0.19
    AS MWSt, Gesamtzielverkaufspreis + Gesamtzielverkaufspreis
    * 0.19 AS Bruttoverkaufspreis
FROM zielvkp_20370;
```

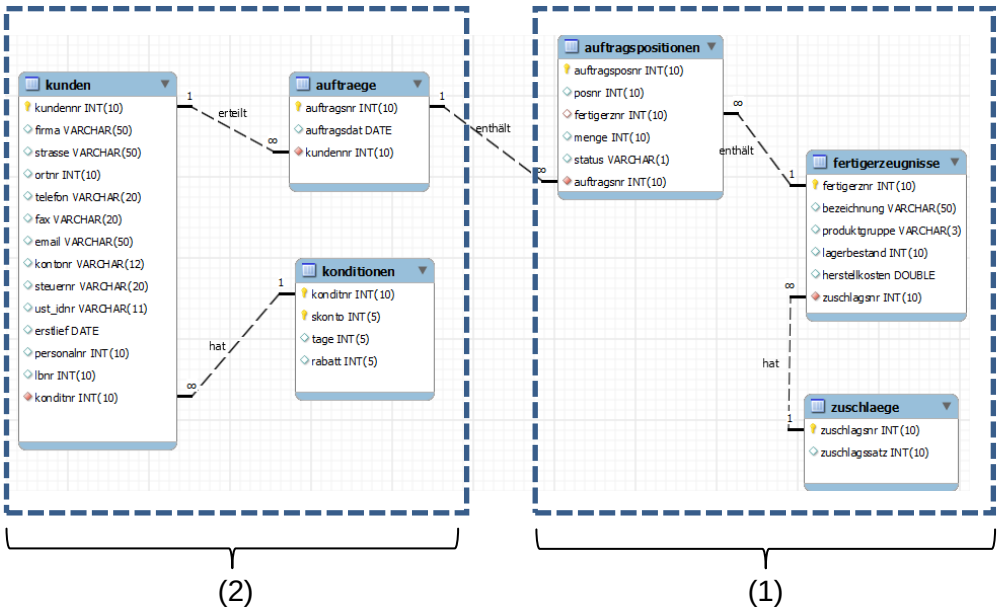
Abfrage 11a Die Abfrage zum Ermitteln des Bruttoverkaufspreises (Abfrage 10b) ist Ausgangspunkt zum Berechnen des Skontobetrages und des Barpreises. Sie wird im Datenbankview *bruttoverkaufspreis* gespeichert.

```
CREATE VIEW bruttoverkaufspreis_20370 AS
    SELECT Gesamtzielverkaufspreis, Gesamtzielverkaufspreis *
    0.19 AS MWSt, Gesamtzielverkaufspreis +
    Gesamtzielverkaufspreis * 0.19 AS Bruttoverkaufspreis
    FROM zielvkp_20370;
```

Abfrage 11b Mit Hilfe des Datenbankviews *bruttoverkaufspreis* und den Tabellen *auftraege, kunden* und *konditionen* werden der Skontobetrag und der Barpreis ermittelt:

```
SELECT Bruttoverkaufspreis, skonto, Bruttoverkaufspreis * skonto/
    100 AS Skonto, Bruttoverkaufspreis - (Bruttoverkaufspreis *
    Skonto / 100) AS Barpreis
FROM bruttoverkaufspreis_20370, auftraege, kunden, konditionen
WHERE auftraege.kundennr = kunden.kundennr
AND kunden.konditnr = konditionen.konditnr
AND auftraege.auftragsnr = 20370;
```

Erläuterung:



(1) Der Datenbankview *bruttoverkaufspreis_20370* basiert auf den Werten der Auftragspositionen (siehe *Abfrage 7*). In dieser Abfrage sind für den Auftrag mit der Auftragsnummer 20370 die Attribute aus den Tabellen *auftrags-positionen, fertigerzeugnisse* und *zuschlaege* für den View berücksichtigt.

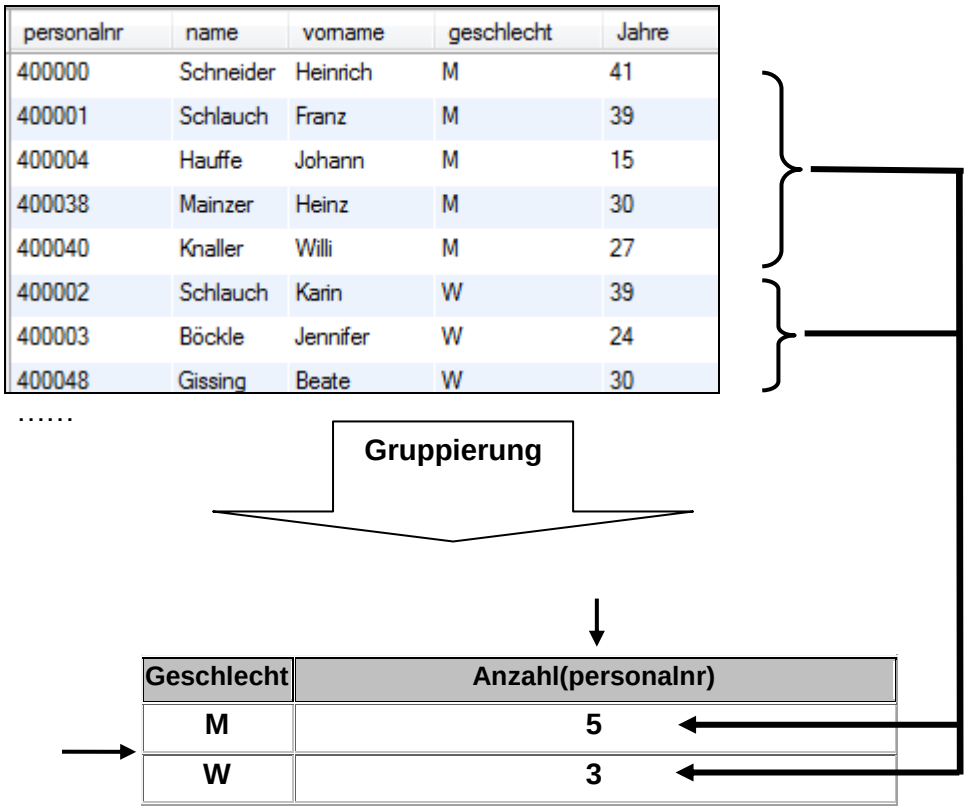
(2) Um noch die Konditionen für den Kunden, der den Auftrag erteilt hat, verwenden zu können, muss für den Auftrag 20370 der Skontosatz (Attribut *skonto*) verwendet werden. Dazu werden die Tabellen *kunden* und *konditionen* benötigt.

1.2.7 Auswahlabfragen mit Gruppierung

Situation:

Um die Bindung der Beschäftigten an die Mücke & Partner OHG einschätzen zu können, möchte die Geschäftsleitung eine Information darüber, wie viele Beschäftige 15 und mehr Jahre im Unternehmen sind. Die Zusammenstellung soll nach dem Geschlecht getrennt sein.

Zur Veranschaulichung sollen zunächst alle Beschäftigten aufgelistet werden:



Erläuterung

Zur Veranschaulichung werden zunächst alle Beschäftigte mit einer Betriebszugehörigkeit von 15 Jahren und mehr, sortiert nach dem Geschlecht, aufgelistet.

SQL-Anweisung:

```
SELECT personalnr, name, vorname, geschlecht, YEAR(NOW()) - YEAR(eintritt) AS Jahre
FROM personal
WHERE YEAR(NOW()) - YEAR(eintritt) >= 15
ORDER BY geschlecht;
```

Um die Daten zusammenfassen (gruppieren) zu können muss festgelegt werden,

(1) nach welchem Attribut gruppiert werden soll, hier ist es das Attribut *geschlecht;*

(2) welches Attribut ausgewertet (gezählt, aufsummiert o. ä) werden soll. Im vorliegenden Beispiel sollen die Mitarbeiter jeweils gezählt werden. Um Fehler möglichst auszuschließen, verwendet man zum Zählen das Primärschlüsselattribut, weil es nie ohne Inhalt ist;

(3) welche Datensätze beim Zählen berücksichtigt werden sollen. Im vorliegenden Beispiel sollen nur die weiblichen beziehungsweise männlichen Beschäftigten gezählt werden, die 15 Jahre oder mehr Betriebszugehörigkeit vorweisen können.

SQL-Anweisung:

```
SELECT geschlecht, COUNT(personalnr) AS Anzahl
FROM personal
WHERE YEAR(NOW()) - YEAR(eintritt) >= 15
GROUP BY geschlecht;
```

Übungsbeispiele zu Gruppierungsabfragen

Abfrage 12 Die Vertriebsleitung wünscht eine Aufstellung, in der die Auftragssummen (Umsätze) der einzelnen Kunden für das Kalenderjahr 2013 zusammengefasst sind.
Die Aufstellung soll nach den Auftragssummen absteigend sortiert sein, so dass der umsatzstärkste Kunde am Anfang steht.

Auszug aus der Ergebnistabelle:

kundennr	firma	Gesamt
270050	Möbel-Rau	978245.4924999997
270019	L'ALTRA GALLERIA GmbH	743119.7549999998
270026	SCHERER Einrichtungen GmbH	732075.155
270028	Wohnen und Design	728121.6400000001
270038	Möbel Hirschlinger OHG	685680.18
270041	Moll Stilmöbel	529372.04
270039	Rustikalmöbel GmbH	446208.98999999993

Abfrage 13 Es sollen nur noch die Kunden angezeigt werden, deren Umsatz 500.000,00 Euro erreicht oder übersteigt.

Abfrage 14 Die Vertriebsleitung der Mücke & Partner OHG wünscht eine Aufstellung, in der die Auftragssummen (Monatsumsätze) eines jeden Monats des Kalenderjahres 2013 aufgeführt sind.

Lösungsvorschläge

Abfrage 12

```
SELECT kunden. kundennr, firma, SUM(menge * (herstellkosten +
    herstellkosten * zuschlagssatz / 100)) AS Gesamt
FROM kunden, auftraege, auftragspositionen, fertigerzeugnisse,
    zuschlaege
WHERE kunden.kundennr = auftraege.kundennr
AND auftraege.auftragsnr = auftragspositionen.auftragsnr
AND auftragspositionen.fertigerznr=fertigerzeugnisse.fertigerznr
AND fertigerzeugnisse.zuschlagsnr = zuschlaege.zuschlagsnr
AND YEAR(auftragsdat)=2013
GROUP BY kunden.kundennr
ORDER BY Gesamt DESC;
```

Abfrage 13 Die Abfrage 12 wird um die „**HAVING**“-Klausel erweitert:

```
SELECT kunden.kundennr, firma, SUM(menge * (herstellkosten +
    herstellkosten * zuschlagssatz / 100)) AS Gesamt
FROM kunden, auftraege, auftragspositionen, fertigerzeugnisse,
    zuschlaege
WHERE kunden.kundennr = auftraege.kundennr
AND auftraege.auftragsnr = auftragspositionen.auftragsnr
AND auftragspositionen.fertigerznr=fertigerzeugnisse.fertigerznr
AND fertigerzeugnisse.zuschlagsnr= zuschlaege.zuschlagsnr
AND YEAR (auftragsdat)=2013
GROUP BY kunden.kundennr
HAVING Gesamt >=500000
ORDER BY Gesamt DESC ;
```

Erläuterung: Die ***HAVING-Klausel*** wird in Verbindung mit der Gruppierungsabfrage (*GROUP BY*) verwendet.
Im Unterschied zur **WHERE-Klausel**, mit der die Zeilen ausgewählt werden, die in die Gruppierung eingehen, wird mit der **HAVING-**Klausel **bestimmt, welche Gruppen ausgewählt werden** und in die Ergebnismenge eingehen.

Abfrage 14

```
SELECT MONTH(auftragsdat) AS Monat, SUM(menge * (herstellkosten +
    herstellkosten * zuschlagssatz / 100)) AS Gesamt
FROM auftraege, auftragspositionen, fertigerzeugnisse, zuschlaege
WHERE auftraege.auftragsnr = auftragspositionen.auftragsnr
AND auftragspositionen.fertigerznr=fertigerzeugnisse.fertigerznr
AND fertigerzeugnisse.zuschlagsnr = zuschlaege.zuschlagsnr
AND YEAR(auftragsdat)=2013
GROUP BY MONTH(auftragsdat)
ORDER BY MONTH(auftragsdat);
```

1.2.8 Auswahlabfragen mit Unterabfragen

Mit Unterabfragen werden andere SQL-Auswahlabfragen in eine Hauptabfrage eingebunden. Das/die Ergebniss/e der Unterabfrage werden in der Hauptabfrage sofort verwendet.

Unterabfragen können auch in *DELETE*-, *UPDATE*- oder *INSERT*-Anweisungen verwendet werden.

Grundsätzlich können Unterabfragen

- einen Wert,
- mehrere Werte (Liste)

als Ergebnis für die Weiterverwendung in der Hauptabfrage liefern.

Beim Anwenden von Unterabfragen ist zu beachten:

- Die Unterabfrage wird in runde Klammer gesetzt.
- Liefert die Unterabfrage einen einzelnen Datensatz als Rückgabewert, können Vergleichsoperatoren =, >, >, >=, <=, !=; <> verwendet werden (siehe nachfolgendes Beispiel 1).
- Bei mehreren Datensätzen als Rückgabewert verwendet man die Mengenoperatoren *ALL, ANY, IN, EXISTS* (siehe nachfolgendes Beispiel 2). Die Mengenoperatoren *ALL* und *ANY* können nur zusammen mit einem Vergleichsoperator verwendet werden, z. B. *< ALL.*
- *ORDER BY* ist in einer Unterabfrage nicht zulässig.

Beispiel 1: Unterabfragen mit einem Rückgabewert

Situation:

Der Betriebsrat der Mücke & Partner *OHG* beabsichtigt die Gehaltsstruktur der Firma zu untersuchen. Er möchte eine Aufstellung, aus der ersichtlich ist, welche Mitarbeiterinnen oder Mitarbeiter weniger als 20% unter dem Durchschnittsgehalt aller Mitarbeiter verdienen.

In der Aufgabenstellung sind zwei Fragen enthalten:

- Wie hoch ist das durchschnittliche Gehalt aller Mitarbeiter?
- Welche Mitarbeiter erhalten weniger als das um 20 % verminderte Durchschnittsgehalt.

Lösung mit zwei Einzelabfragen:

1) Durchschnittsgehalt aller Mitarbeiter ermitteln.

```
SELECT AVG(gehalt) AS Durchschnittsgehalt
FROM personal;
```

Das Ergebnis dieser Abfrage:

Durchschnittsgehalt
2853.044872

Das Ergebnis dieser Abfrage wird als Bedingung in die WHERE-Klausel der zweiten Abfrage übernommen.

2) Mitarbeiterinnen/Mitarbeiter auswählen, die 20% oder mehr unter dem Durchschnittsgehalt erhalten.

```
SELECT personalnr, name, vorname, geschlecht, gehalt
FROM personal;
WHERE gehalt < 0.8 * 2853.04;
```

Lösung mit einer Unterabfrage:

```
SELECT personalnr, name, vorname, geschlecht, gehalt
FROM personal;
WHERE gehalt < 0.8 *   (SELECT AVG(gehalt) AS Durchschnittsgehalt
                       FROM personal);
```

Die WHERE-Klausel der Abfrage wird durch eine eigene Abfrage (= Unterabfrage) ermittelt.
Die Unterabfrage muss zwischen Klammern stehen.

Beispiel 2: Unterabfragen mit mehreren Rückgabewerten

Situation:

Der Betriebsrat der Mücke & Partner OHG beabsichtigt die Gehaltsstruktur der Firma zu untersuchen. In der Kostenstelle mit der Kostenstellennummer ‘05‘ sind die Hilfskräfte zusammengefasst, deren Gehaltsniveau gegenüber der Gesamtbelegschaft niedriger ist.
Es soll nun ermittelt werden, ob es in den anderen Kostenstellen (Attribut *kostennr*) Mitarbeiterinnen oder Mitarbeiter gibt, die das gleiche Gehalt beziehen, wie die Mitarbeiter/Mitarbeiterinnen in der Kostenstelle ‚05‘.
Die Aufstellung soll die Attribute *personalnr, name, vorname, gehalt* und *kostennr* umfassen. Sie soll nach dem Attribut *gehalt* und bei Gleichheit nach dem Attribut *kostennr* sortiert werden.

Lösung:

1. Formulieren der Unterabfrage: Gehälter der Mitarbeiter der Kostennr 05 auflisten:

```
SELECT gehalt
FROM personal
WHERE kostnr = '05';
```

Die Abfrage liefert als Ergebnis die nebenstehende Liste.

gehalt
4400.00
1900.00
4950.00
2050.00
2125.00
1900.00
1900.00
6100.00

Wenn aus den anderen Kostenstellen diejenigen Mitarbeiter/innen angezeigt werden sollen, deren Gehalt auch in der Liste der Mitarbeiter/innen der Kostenstellennummern 05 vorkommt, muss die Unterabfrage in die Hauptabfrage eingebunden werden:

```
SELECT name, gehalt, kostnr
FROM personal
WHERE gehalt IN ( SELECT gehalt
                  FROM personal
                  WHERE kostnr = '05')
ORDER BY gehalt, kostnr;
```

Ergebnis:

name	gehalt	kostnr
Kolesnik	1900.00	04
Mainzer	1900.00	05
Schlauch	1900.00	05
Berger	1900.00	05
Sautter	1900.00	06
Dreizler	1900.00	06

Bemerkung: Der Mengenoperator *IN* bedeutet, dass ein Datensatz nur angezeigt wird, wenn in der Ergebnisliste der Unterabfrage der Vergleichswert übereinstimmt.

Schlauch	4400.00	05
Schweizer	4950.00	05
Hauffe	6100.00	05

Übungsbeispiel

Abfrage 15 Welche Kunden hatten im Kalenderjahr 2013 einen überdurchschnittlich hohen Umsatz?

Hinweis: In der Abfrage 12 wurde eine Auflistung erstellt, in der die Auftragssummen (Umsätze) der einzelnen Kunden ausgewiesen werden.

Übernehmen Sie diese Abfrage in einen Datenbankview mit dem Namen *umsatzuebersicht_2013* und wählen Sie anschließend aus dem Datenbankview die Kunden mit überdurchschnittlichem Umsatz aus.

Lösungsvorschlag

(1) Datenbank-View *umsatzübersicht_2013* erstellen

```
CREATE VIEW umsatzuebersicht_2013 AS
    SELECT kunden.kundennr, firma, SUM(menge * (herstellkosten +
          herstellkosten * zuschlagssatz / 100)) AS Gesamt
    FROM kunden, auftraege, auftragspositionen, fertigerzeugnisse,
          zuschlaege
    WHERE kunden.kundennr = auftraege.kundennr
    AND auftraege.auftragsnr = auftragspositionen.auftragsnr
    AND auftragspositionen.fertigerznr = fertigerzeugnisse.fertigerznr
    AND fertigerzeugnisse.zuschlagsnr = zuschlaege.zuschlagsnr
    AND YEAR(auftragsdat) = 2013
    GROUP BY kunden.kundennr
    ORDER BY Gesamt DESC;
```

(2) Aus dem Datenbank-View umsatzuebersicht_2011 die Kunden auswählen, deren Gesamtumsatz über dem durchschnittlichen Umsatz liegt.

```
SELECT kundennr, firma, gesamt
FROM umsatzuebersicht_2013
WHERE Gesamt > ( SELECT AVG(Gesamt)
                 FROM umsatzuebersicht_2013);
```

Exkurs: SQL und Boolesche Algebra

SQL ist eine an der Mengenlehre ausgerichtete Sprache. Es gelten somit die Axiome der Booleschen Algebra. Bei Kenntnis dieser Gesetze können Boolesche Ausdrücke zum Teil sehr verkürzt und Abfragen daher wesentlich beschleunigt werden. Ebenfalls kann darauf hingewiesen werden, dass eine Abfrage über das Komplement wesentlich einfacher konstruiert werden kann.[1]

i) Einige Axiome der Booleschen Algebra:

- Mit A, B und C sind Mengen definiert.
- Mit "UND" (AND - $\wedge$) werden deren Schnittmengen (A $\wedge$ B),
- mit "ODER" (OR - $\vee$) werden deren Vereinigungsmengen definiert (A $\vee$ B);
- "NICHT" (NOT - $\neg$) bedeutet, dass alle Elemente außerhalb einer Menge angesprochen sind ($\neg$ A).

Beispiel: Die Menge "A" umfasst alle Vereinsmitglieder aus Stuttgart; dann gehören zur Menge "nicht_A" alle Vereinsmitglieder, die nicht aus Stuttgart kommen.

Das Verschmelzungsgesetz:

- A $\vee$ (A $\wedge$ B) = A Verschmelzung der Vereinigung
- A $\wedge$ (A $\vee$ B) = A Verschmelzung des Durchschnitts

Beispiel in SQL (auf Basis der Tabelle *fertigerzeugnisse*)

```
1.  Teil: Verschmelzung der Vereinigung
          SELECT *
          FROM fertigerzeugnisse
          WHERE herstellkosten < 500
          OR (herstellkosten < 500 AND produktgruppe = "I");

    Abfrage kann ersetzt werden durch:
          SELECT *
          FROM fertigerzeugnisse
          WHERE herstellkosten < 500;


2.  Teil: Verschmelzung des Durchschnitts
          SELECT *
          FROM fertigerzeugnisse
          WHERE herstellkosten < 500
          AND (herstellkosten < 500 OR produktgruppe = "I");

    Abfrage kann ersetzt werden durch:
          SELECT *
          FROM fertigerzeugnisse
          WHERE herstellkosten < 500;
```

[1] Die folgenden Ausführungen sind als Hintergrundwissen für Lehrkräfte zu verstehen und gehen über den Lehrplan hinaus.

Gesetze von De Morgan:

- $\neg (A \wedge B) = \neg A \vee \neg B$
- $\neg (A \vee B) = \neg A \wedge \neg B$

Beispiel in SQL

```
1. Teil der Gesetze von De Morgan:
        SELECT *
        FROM fertigerzeugnisse
        WHERE NOT(zuschlagsnr = 3 OR zuschlagsnr = 5);

   Abfrage kann ersetzt werden durch:
        SELECT *
        FROM fertigerzeugnisse
        WHERE NOT zuschlagsnr = 3 AND NOT zuschlagsnr = 5;


2. Teil der Gesetze von De Morgan:
        SELECT *
        FROM fertigerzeugnisse
        WHERE NOT (zuschlagsnr = 3 AND lagerbestand<100);

   Abfrage kann ersetzt werden durch:
        SELECT *
        FROM fertigerzeugnisse
        WHERE NOT zuschlagsnr = 3 OR NOT lagerbestand<100;
```

ii) Wahrheitstabelle

Wahrheitstabellen werden verwendet, um boolesche Funktionen zu definieren, darzustellen und um ihre Aussagenlogik zu veranschaulichen. Sie stellt das Ergebnis einer booleschen Funktion in Abhängigkeit der möglichen Kombinationen der Werte der Eingangsvariablen dar.
Die folgende Wahrheitstabellen zeigen beispielsweise die Gültigkeit der de Morgan'schen Gesetze:

- $\neg (A \wedge B) = \neg A \vee \neg B$

A	**B**	**$\neg (A \wedge B)$**	**$\neg A \vee \neg B$**
0	0	1	1
1	0	1	1
0	1	1	1
1	1	0	0

- $\neg (A \vee B) = \neg A \wedge \neg B$

A	**B**	**$\neg (A \vee B)$**	**$\neg A \wedge \neg B$**
0	0	1	1
1	0	0	0
0	1	0	0
1	1	0	0