Informationssysteme (Übung)

Übung 1

Aufgabe 1

Datenbanksystem: Datenbank + Datenbankmanagement

Aufgabe 2

Struktur

```
CREATE TABLE wein (
ID INT,
Art VARCHAR(30),
Lage VARCHAR(30),
Jahr INT,
Alc NUMERIC(3,1));
```

Daten:

```
INSERT INTO wein VALUES
  (2, 'Chardonnay', 'Buena Vista',2003,11.0);
INSERT INTO wein VALUES
  (3, 'Chardonnay', 'Louis Martini', 2004, 11.5),
  (.....);
```

Ausgabe:

```
SELECT * FROM wein;
```

Aufgabe 3

a) i)

```
SELECT * FROM wein WHERE Alc >= 11.5;
```

ii)

```
SELECT Lage FROM wein WHERE Art = 'Riesling';
```

iii)

```
SELECT ID, Art FROM wein
```

```
    WHERE Jahr >= 2002 AND Jahr <= 2004;
```

```
SELECT ID, Art FROM wein
    WHERE Jahr BETWEEN 2002 AND 2004;
```

iv)

```
SELECT Alc FROM wein
    WHERE Art = 'Weissburgunder' AND Lage = 'Mirassou';
```

v)

```
SELECT * FROM wein WHERE Art = 'Chardonnay';
```

b)

```
ALTER TABLE wein ADD Vorrat INT;
UPDATE wein SET Vorrat = 5
    WHERE ID = 16;
UPDATE wein SET Vorrat = 3
    WHERE Art = 'Pino Noir' AND Lage = 'St. Jean';
```

c)

```
DELETE FROM wein WHERE Jahr = 1997;
```

Übung 2

Aufgabe 1

Struktur:

```
CREATE TABLE Artikel (
  Artikelnr INT,
  Bezeichnung VARCHAR(30),
  Verkaufspreis NUMERIC(5,2),
  Einkaufspreis NUMERIC(5,2));
```

<note tip>Wenn Groß / Kleinschreibung eine Rolle spielen soll, müssen die Benennungen in „“ notiert werden, z.B. create table „Artikel“.</note> Daten:

```
INSERT INTO Artikel VALUES
  (95, 'Kamm',1.50,0.80);
INSERT INTO Artikel VALUES
  (97, 'Kamm',0.99,0.75),
  (507, 'Seife',3.95,2.40);
```

Ausgabe der Daten:

```
SELECT * FROM Artikel;
```

Aufgabe 2

a) i)

```
SELECT Artikelnr FROM Artikel
    WHERE Bezeichnung = 'Zwieback';
```

ii)

```
SELECT * FROM Artikel WHERE Verkaufspreis > 25;
```

iii) Mehrere Möglichkeiten

```
SELECT Artikelnr, Bezeichnung FROM Artikel
    WHERE (Einkaufspreis > 1) AND (Einkaufspreis <= 15);
SELECT Artikelnr, Bezeichnung FROM Artikel
    WHERE Einkaufspreis BETWEEN 1 AND 15;
SELECT Artikelnr, Bezeichnung FROM Artikel
    WHERE NOT Einkaufspreis < 1 AND NOT Einkaufspreis > 15);
```

iv)

```
SELECT Artikelnr, Verkaufspreis FROM Artikel
    WHERE Bezeichnung = 'Herrenhose' AND Verkaufspreis < 25;
```

v)

```
SELECT * FROM Artikel
    WHERE Verkaufspreis - Einkaufspreis >= 2;
/*! Alternative Methode: */
SELECT *, Verkaufspreis - Einkaufspreis AS Gewinn FROM Artikel
    WHERE Verkaufspreis - Einkaunfspreis >= 2;
```

Zusatz: Gib alle Artikelnummern und Bezeichnungen aus, bei denen das Unternehmen mehr als 10% Gewinn macht.

```
SELECT Artikelnr, Bezeichnung FROM Artikel
    WHERE Verkaufspreis >= 1.1 * Einkaufspreis;
```

b)

```
ALTER TABLE Artikel ADD Hersteller VARCHAR(30);
UPDATE Artikel SET Hersteller = 'Gromann Moden'
    WHERE Artikelnr = 2045 OR Artikelnr = 2046;
/*! Alternative Methoden: */
UPDATE Artikel SET Hersteller = 'Gromann Moden'
```

```
    WHERE Artikelnr BETWEEN 2045 AND 2046;
UPDATE Artikel SET Hersteller = 'Gromann Moden'
    WHERE Artikelnr IN (2045,2046);
```

c)

```
DELETE FROM Artikel WHERE Bezeichnung = 'Kamm';
```

Übung 3

Aufgabe 1

b)
i) Alle Spalten sollen ausgegeben werden, bei denen das D bei Tab 1 kleiner ist als das D bei Tab 1. Tabelle hat so viele Spalten, wie beide Tabellen zusammen und so viele Reihen wie Zeilen von Tabelle 1 mal Zeilen von Tabelle 2.

a	b	c	d	e	f	d
1	2	3	4	4	5	6
1	2	3	4	7	8	9
1	2	3	4	10	11	12
5	6	7	8	7	8	9
5	6	7	8	10	11	12

ii) Tabelle mit 3 Spalten aus beiden Tabellen, bei denen B = F ist oder A+C kleiner gleich als D von Tabelle 2. <note tip>Where Bedingung bezieht sich auf die gesamte Zeile bei einem Test.</note>

d	e	f
4	1	2
4	4	5
4	7	8
4	10	11
8	10	11

iii) Tabelle mit 3 Spalten aus beiden Tabellen, bei denen C>F ist.

a	b	c
1	2	3
5	6	7
5	6	7
9	10	11
9	10	11
9	10	11

Aufgabe 2

a)

```
CREATE TABLE Abteilungen
  (Abteilungsname VARCHAR(30),
  Stockwerk VARCHAR(2),
  Abteilungsleiter VARCHAR(30));
INSERT INTO Abteilungen VALUES
  ('Kosmetik', 'I', 'Franz Ortner');
INSERT INTO Abteilungen VALUES
  ('Lebensmittel', 'EG', NULL);
/*! Alternative Methode: */
INSERT INTO Abteilungen (Abteilungsname, Stockwerk) VALUES
  ('Lebensmittel', 'EG');
```

```
CREATE TABLE Bestand (Abteilungsname VARCHAR(30), Artikelnr INT, Vorrat
INT);
/*! Zitat Röder: Warum gibt´s die Seife in der Lebensmittelabteilung?
Naja, jetzt weiß ich auch, warum er Schaum vor´m Mund hatte.*/
INSERT INTO Bestand VALUES
  ('Kosmetik', 507, 20),
  (...),
  ('Textilien', 230, 4);
```

b)

```
UPDATE Artikel SET Verkaufspreis = Verkaufspreis-2
    WHERE Artikelnr = 2045;
UPDATE Bestand SET Vorrat = Vorrat-4
    WHERE Artikelnr = 507 AND Abteilungsname = 'Kosmetik';
UPDATE Bestand SET Vorrat = Vorrat-2
    WHERE Artiklenr = 507 AND Abteilungsname = 'Lebensmittel';
UPDATE Abteilungen SET Abteilungsleiter = 'Klaus Fretter'
    WHERE Abteilungsname = 'Lebensmittel';
```

c)Joins
i)

```
SELECT Vorrat FROM Bestand, Artikel
    WHERE (Artikel.Artikelnr = Bestand.Artikelnr) AND (Bezeichnung =
'Zwieback');
```

ii)

```
SELECT Stockwerk FROM Artikel, Abteilungen, Bestand
    WHERE (Artikel.Artikelnr = Bestand.Artikelnr) AND
    (Bestand.Abteilungsname = Abteilungen.Abteilungsname)
    AND (Bezeichnung = 'Räucherlachs');
```

iii)

```
SELECT Abteilungsleiter FROM Artikel, Abteilungen, Bestand
    WHERE (Artikel.Artikelnr = Bestand.Artikelnr)
    AND (Bestand.Abteilungsname = Abteilungen.Abteilungsname)
    AND (Bezeichnung = 'Herrenhose');
/*! Ausgabe: Monika Siehl, Monika Siehl */
SELECT Abteilungsleiter FROM Artikel, Abteilungen, Bestand
    WHERE (Artikel.Artikelnr = Bestand.Artikelnr)
    AND (Bestand.Abteilungsname = Abteilungen.Abteilungsname)
    AND (Bezeichnung = 'Seife');
```

Übung 4

Aufgabe 1

a)

```
CREATE INDEX IndVP ON Artikel (Verkaufspreis);
```

Befehl erstellt eine Tabelle mit zwei Spalten, wovon eine die genannte Spalte sortiert ausgibt - hier Verkaufspreis. Die zweite Spalte der neuen Tabelle legt einen Verweis auf die entsprechende Zeile in der genannten Tabelle an, hier Artikel. Wirkung des Befehls bleibt erhalten.
Suchen mit select werden durch Indexe erheblich beschleunigt. Maximale Schritte bis gefunden: Zweier Logarithmus der Anzahl aller Zeilen. b)

```
CREATE INDEX IndBezEin ON Artikel (Bezeichnung, Einkaufspreis);
```

Bei zwei Spalten wird erst nach der ersten genannten Spalte und dann nach der zweiten genannten sortiert. c)

```
DROP INDEX IndVP;
DROP INDEX IndBezEin;
```

delete wird für die Daten eingesetzt, drop für die Struktur der Datenbank.

```
CREATE UNIQUE INDEX IndBez ON Artikel (Bezeichnung);
/*! führt zu Fehlermeldung, da die Items bei Bezeichnung nicht eindeutig
sind (sie kommen mehrfach vor, z.B. Herrenhose */
```

Aufgabe 2

a)

```
CREATE VIEW Angebote AS
  SELECT Bezeichnung, Verkaufspreis FROM Artikel
    WHERE Verkaufspreis < 1.35 * Einkaufspreis;
```

b)

```
UPDATE Artikel SET Bezeichnung = 'Cordhose'
    WHERE Artikelnummer = 2046;
SELECT * FROM Angebote;
```

Ergebnis:

Bezeichnung	Verkaufspreis
Zwieback	1.20
Sommerkleid	94.20
Cordhose	20.00

View erstellt eine Schablone für die Tabelle, so dass nur bestimmte Werte angezeigt werden, d.h. Änderungen in der Tabelle werden bei der Ausgabe des Views berücksichtigt. c) mit \c [Benutzername] werden die Datenbanken eines anderen Nutzers angezeigt, allerdings können in der Regel keine SQL-Befehle auf einer fremden Datenbank ausgeführt werden, außer es existieren bestimmte Zugriffsrechte. Hier wird des Recht select auf eine View von vorhin einem anderen Nutzer gewährt.

```
GRANT SELECT ON Angebote TO public;
/*! Weitere Möglichkeit um Nutzern einen Zugriff zu erlauben: */
GRANT SELECT ON Angebote TO roeder, maier;
/*! Weitere Möglichkeit um Gruppen an Benutzern einen Zugriff zu erlauben:
*/
GRANT SELECT ON Abgebote TO GROUP projekt;
/*! Recht wieder entfernen */
REVOKE SELECT ON Angebote FROM public;
```

d)

```
CREATE VIEW Billig AS
  SELECT * FROM Angebote WHERE Verkaufspreis <= 25;
```

Es ist möglich eine View auf eine View zu erstellen. e)

```
DROP VIEW Angebote;
/*! führt zu Fehler, da die Sicht Billig bereits existiert
und sich diese auf Angebote bezieht, daher: */
DROP VIEW Billig;
DROP VIEW Angebote;
```

Aufgabe 3

a) union vereinigt beide Ergebnistabellen. Anzahl der Spalten und die Datentypen der Ergebnistabellen müssen gleich sein, d.h. semantisch sollten beide Tabellen gleich sein, sonst vermischt sich die Darstellung.

Bezeichnung	Verkaufspreis
Herrenhose	35.50
Cordhose	20.00
Seife	2.40
Zwieback	0.90
Räucherlachs	3.60
Cordhose	17.00

b)

Artikelnummer	Vorrat
2340	4
2046	9
2045	14

c) Order by 2 bezieht sich auf die zweite select Bedingung

Artikelnummer	Bezeichnung	Vorrat
1401	Räucherlachs	6
507	Seife	11
1056	Zwieback	129

Übung 5

Aufgabe 1

a)

```
SELECT COUNT(*) FROM Bestand WHERE Abteilungsname='Lebensmittel';
/*! as anzahl kann noch hinzugefügt werden,
um die Tabellenüberschrift der Ausgabe zu setzen*/
```

b)

```
SELECT MIN(Verkaufspreis), MAX(Verkaufspreis) FROM Artikel;
```

c)

```
SELECT SUM(Vorrat) FROM Bestand, Artikel
    WHERE (Artikel.Artikelnr = Bestand.Artikelnr)
    AND (Bezeichnung='Seife');
```

Aufgabe 2

a) distinct bedeutet, dass nur die verschiedenen Werte von einer Spalte (hier: A) gezählt werden.

count
4

b) group by gruppiert eine Spalte nach Gruppen mit gleichen Werten.

a	count	sum	max	avg
1	1	2	1	2
9	3	12	9	4
3	2	10	3	5
2	1	7	2	7
7	1	4	7	4

c) having min bedeutet, dass nur die Werte einer Spalte ausgeliefert werden können, die eine bestimmte Minimum Bedingung haben. <note tip>„Having“ wird auf Gruppen (group by) angewendet, „where“ wird auf Zeilen angewendet.</note>

a	sum
3	10
2	7
7	4

d) Bedingungen werden folgendermaßen geprüft: erst where, dann group by und dann das having

min	b
7	4

Aufgabe 3

```
UPDATE Artikel SET Bezeichnung='Herrenhose'
    WHERE Bezeichnung='Cordhose';
```

a)

```
SELECT avg(Verkaufspreis) FROM Artikel
    WHERE Bezeichnung='Herrenhose';
```

b)

```
SELECT Abteilungsname, SUM(Vorrat) FROM Bestand GROUP BY Abteilungsname;
```

c)

```
SELECT Abteilungsname FROM Bestand
    GROUP BY Abteilungsname HAVING SUM(Vorrat)<100;
```

d)

```
SELECT Abteilungsname, MIN(Verkaufspreis), MAX(Verkaufspreis) FROM Bestand,
Artikel
```

```
    WHERE (Artikel.Artikelnr = Bestand.Artikel)
    GROUP BY Abteilungsname;
```

e)

```
SELECT Artikelnr, SUM(Vorrat) FROM Bestand
    GROUP BY Artikelnr HAVING SUM(Vorrat)>=30;
```

Aufgabe 4

```
CREATE temp TABLE Neu AS SELECT * FROM Artikel WHERE FALSE;
/*! where false erstellt eine leere Tabelle mit
dem definierten Schema via as select * from */
INSERT INTO Neu VALUES
  (2110, 'Cordhose', 33.00, 20.00, 'Gromann Moden'),
  (...);
```

a)

```
INSERT INTO Artikel SELECT * FROM Neu;
```

b)

```
/*! i) */
SELECT Artikelnr, Einkaufspreis FROM Artikel
    WHERE Bezeichnung LIKE '%hose'
/*! like betrachtet nach Vorkommen, % meint die Stelle mit beliebigen
Zeichen*/
/*! ii)*/
SELECT * FROM Artikel WHERE Bezeichnung LIKE '%eife';
/*! iii) */
SELECT Artikelnr, Bezeichnung FROM Artikel
    WHERE Hersteller LIKE 'Ma%erhofer';
```

Übung 6

Aufgabe 1

i) exists testet, ob das Ergebnis einer Abfrage mindestens ein Ergebnis liefert, daher meist mit select * fortgesetzt. Das Ergebnis von exists ist wie true / false zu sehen, d.h. ist die Ergebnistabelle leer oder nicht. Wenn nicht leer, wird das vor exists geschrieben, wenn nicht, wird gar nichts geschrieben.

a	c
1	3
5	7

ii) Tab2 kann weggelassen werden, wenn sich in der Abfrage ein korrektes from befindet, dann wird

das automatisch berücksichtigt. not exists gibt nur ein Ergebnis aus, wenn die Ergebnistabelle leer ist.

b
2

iii) in bedeutet, ob etwas in einer Liste vorkommt. Kein select * benötigt, da pro Zeile getestet wird und das Ergebnis in der In Condition immer gleich ist, aber es wird der gleiche Datentyp in allen überprüften Tabellen benötigt.

e	d
10	12

iv) in sucht in einer Liste von Werten, = verlangt, dass dabei nur ein Wert in der inneren Abfrage herauskommt.

d
8

Aufgabe 2

i) Nicht jeder Join kann als geschachtelte Select-Anweisung formuliert sein. Das geht nur, wenn erfüllt ist: Inhalt der zweiten Tabelle wird nicht benötigt.

```
SELECT Bezeichnung, Verkaufspreis FROM Artikel, Bestand
    WHERE Artikel.Artikelnr = Bestand.Artikelnr
    AND Abteilungsname = 'Textilien';
/*! Andere Möglichkeit */
SELECT Berezichnung, Verkaufspreis FROM Artikel
    WHERE Artikelnr
    IN (SELECT Artikelnr FROM Bestand WHERE Abteilungsname = 'Textilien');
```

ii)

```
SELECT Bezeichnung FROM Artikel
    WHERE Artikelnr IN (SELECT Artikelnur FROM Bestand WHERE Abteilungsname
    IN (SELECT Abteilungsname FROM Abteilungen WHERE Stockwerk = 'EG'));
```

iii) Trick: Umformulieren, d.h. es gibt kein Element, für das nicht gilt. Wenn steht „für alle“ immer umdrehen.

```
SELECT Abteilungsleiter FROM Abteilungen
    WHERE NOT EXISTS
    (SELECT * FROM Bestand WHERE Abteilungen.Abteilungsname = Abteilungsname
AND Vorrat >= 100);
```

iv)

```
SELECT Abteilungsleiter FROM Abteilungen
    WHERE NOT EXISTS
    (SELECT * FROM Bestand WHERE Abteilungen.Abteilungsname = Abteilungsname
```

```
AND Vorrat <= 100);
```

Aufgabe 3

a)

```
/*! geschachteltes Select */
SELECT Bezeichnung, Artikelnr FROM Artikel
    WHERE Verkaufspreis <=
    (SELECT Verkaufspreis FROM Artikel WHERE Artikelnummer = 2045);
/*! Join (Self-Join mit Alias-Namen)*/
SELECT FIRST.Bezeichnung, FIRST.Artikelnr FROM Artikel FIRST, Artikel SECOND
    WHERE (FIRST.Verkaufspreis <= SECOND.Verkaufspreis)
    AND (SECOND.Artikelnr = '2045");
```

b)

```
/*! geschachteltes Select */
SELECT Artikelnr FROM Bestand
    WHERE Abteilungsname IN
    (SELECT Abteilungsname FROM Bestand WHERE Vorrat >= 20);
/*! Join */
SELECT B1.Artikelnr FROM Bestand B1, Bestand B2
    WHERE (B1.Abteilungsname = B2.Abteilungsname)
    AND (B2.Vorrat >= 20);
```

Übung 7

Aufgabe 1

Primärschlüssel ist eine Art unique Index, nur hat eine stärkere Wirkung als der Index. a)

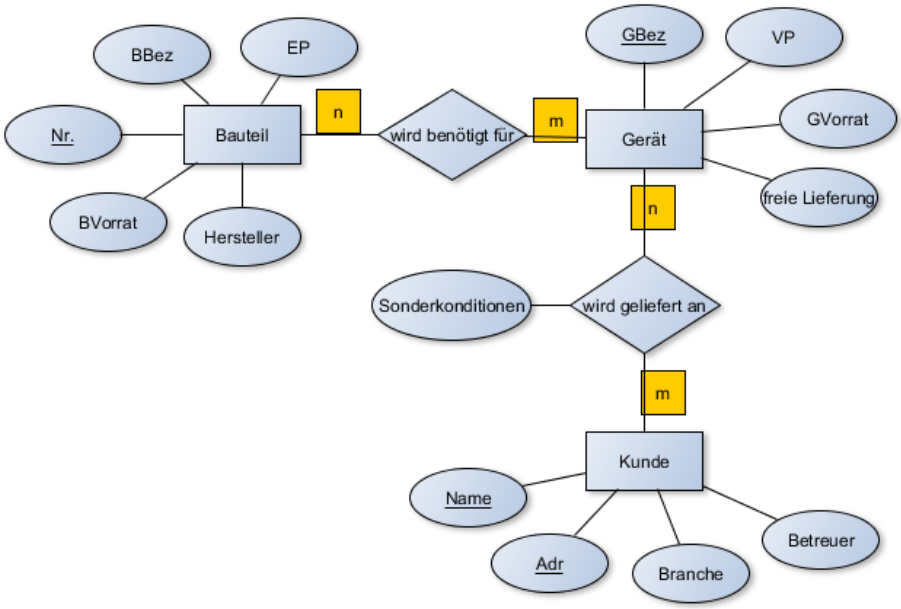
```
ALTER TABLE Artikel ADD PRIMARY KEY (Artikelnr);
ALTER TABLE Abteilungen ADD PRIMARY KEY (Abteilungsname);
ALTER TABLE Bestand ADD PRIMARY KEY (Abteilungsname, Artikelnr);
/*! Primary Key beim Anlegen erstellen: */
CREATE TABLE Artikel
  (Artikelnr INT PRIMARY KEY,
  Bezeichnung VARCHAR(30),
  Verkaufspreis NUMERIC (5,2),
  Einkaufspreis NUMERIC(5,2),
  Hersteller VARCHAR(30));
/*! Primary muss bei mehreren Definition eigens mit
primary key (xy, se) am Ende hinzugefügt werden */
```

b) Erster Befehl verletzt die Primärschlüsselbedingung, da neue Zeile mit gleichem Primärschlüssel hinzugefügt wird, d.h. Fehler ist die Ausgabe.

Zweiter Befehl führt zum selben Problem, ⇒ Fehler.
Dritter Befehl führt auch zum Fehler, da der Primärschlüssel auf Abteilungsname gesetzt ist und damit ein Abteilungsname nur einmal vorkommen darf.
SQL wird ebenfalls meckern, da beim Eintragen kein Wert für die Spalte des Primärschlüssels gesetzt ist, d.h. Primärschlüsselattribute müssen auch einen Wert haben.
Der letzte Befehl funktioniert.

Aufgabe 2

a) n zu m: mehrere Möglichkeiten (nicht eindeutig).
Legende: unterstrichene Attribute bedeuten Primärschlüssel, die organgen Kästchen symbolisieren die möglichen Beziehungen.



b) Legende: unterstrichen bedeutet Primärschlüssel

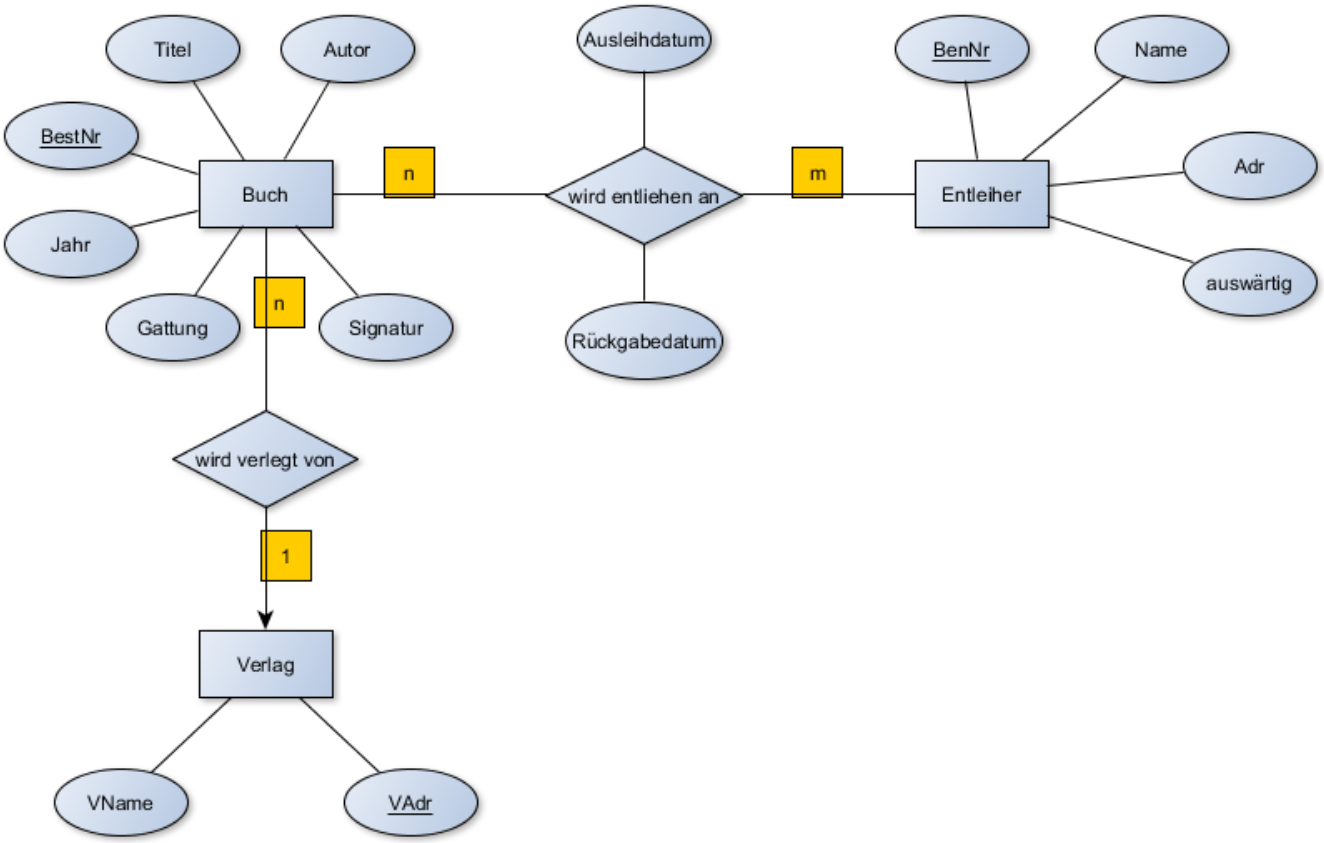
- Bauteil (Nr, BBez, BVorrat, Hersteller, EP
- Gerät (GBez, VP, GVorrat, freie Lieferung)
- Kunde (Name, Adr, Branche, Betreuer)
- wird_benötigt_für (Nr, GBez)
- wird_geliefert_an (GBez, Name, Adr, Sonderkonditionen)

Aufgabe 3

Offtopic:

- Ich habe auch schon mal ein Buch verlegt, aber dann hab ichs wiedergefunden.
- Immer Dienstag Abend habe ich eine weiße Hose.

a)



b) Wenn es eine n zu m Relationship ist, müssen die Primärschlüssel aus beiden Entitytypen verwendet werden, wenn es eine n zu 1 Relationship ist, ist der Primärschlüssel von der n-Beziehung nur relevant.

- Buch (BestellNr, Titel, Autor, Jahr, Gattung, Signatur)
- Entleiher (BenNr, Name, Adr, auswärtig)
- Verlag (VName, VAdr)
- wird_verliehen_an (BestNr, BenNr, Ausleihdatum, Rückgabedatum)
- wird_verlegt_von (BestNr, VAdr)

c) Buch:

- BestNr ⇒ Titel
- BestNr ⇒ Autor
- BestNr ⇒ Jahr
- BestNr ⇒ Gattung
- BestNr ⇒ Signatur
- Signatur ⇒ BestNr
- Signatur ⇒ Titel
- Signatur ⇒ Autor
- Signatur ⇒ Jahr
- Signatur ⇒ Gattung
- Titel, Autor ⇒ BestNr
- Titel, Autor ⇒ Jahr
- Titel ⇒ Gattung

Entleiher:

- BenNr ⇒ Name
- BenNr ⇒ Adr
- BenNr ⇒ auswärtig
- Adr ⇒ auswärtig

Verlag:

- VAdr ⇒ VName

wird_verliehen_an:

- BestNr, Ausleihdatum ⇒ Rückgabedatum
- BestNr, Ausleihdatum ⇒ BenNr
- BestNr, Rückgabedatum ⇒ BenNr
- BestNr, Rückgabedatum ⇒ Ausleihdatum

wird_verlegt_von:

- BestNr ⇒ VAdr

Schlüsselkandidaten

Buch: {BestNr}, {Signatur}, {Titel, Autor}
Entleiher: {BestNr}
Verlag: {VAdr}
wird_verliehen_an: {BestNr, Ausleihdatum}, BestNr, Rückgabedatum}
wird_verlegt_von: {BestNr}

Übung 8

Aufgabe 1

a) Fremdschlüssel, sind Primärschlüssel, die als Attribute in anderen Tabellen vorkommen - kann auch gegenseitig vorkommen. Das ist wichtig wegen der referenziellen Integrität, d.h. dass als Fremdschlüssel definierte Einträge auch Primärschlüssel haben, ob die Datensätze zusammenpassen.

```
ALTER TABLE Bestand ADD FOREIGN KEY (Abteilungsname) REFERENCES Abteilungen;
ALTER TABLE Bestand ADD FOREIGN KEY (Artikelnr) REFERENCES Artikel;
/*! Primary und foreign Key bei create table Befehl, zwei Möglichkeiten */
CREATE TABLE Bestand (Abteilungsname VARCHAR(30) /*! Möglichkeit 1*/
REFERENCES Abteilungen, Artikelnur INT, Vorrat INT, /*! Möglichkeit 2*/
PRIMARY KEY(Abteilungsname, Artikelnr), FOREIGN KEY (Abteilungsname)
REFERENCES Abteilungen, FOREIGN KEY (Artikelnr) REFERENCES Artikel);
```

b) Anwedungsspezifische Integritätsregeln sind Regeln, die bei der Eingabe von Daten erfüllt sein müssen, z.B. alle Artikelnummern müssen 3-stellig sein. Referenzielle Integrität sorgt dafür, dass nur einmal der Check für ein Attribut gemacht werden muss, wenn dieses mit einer Referenz zu einer anderen Tabelle vorkommt.

```
ALTER TABLE Artikel ADD CHECK (Artikelnr >= 100);
ALTER TABLE Artikel ADD CHECK (Verkaufspreis > 0);
ALTER TABLE Artikel ADD CHECK (Einkaufspreis > 0);
ALTER TABLE Artikel ADD CHECK (Verkaufspreis >= Einkaufspreis);
ALTER TABLE Bestand ADD CHECK (Vorrat >= 0);
ALTER TABLE Abteilungen ADD CHECK (Stockwerk IN ('I', 'II', 'EG', 'UG'));
```

c) Strukturelle Integrität beschreibt bei

- der 1. dass jede Tabelle einen primary Key braucht,
- die 2. sagt, dass nur Fremdschlüsselwerte gleich der Primärschlüsselwerte sind und
- die 3. dass die Werte von primary Key Attributen nicht NULL sind.

```
/*! Verstoß gegen Regel 3 */
INSERT INTO Artikel (Bezeichnung, Einkaufspreis) VALUES
  ('Zahnpasta', 1.50);
/*! Verstoß gegen die referenzielle Integrität, Regel 2 */
INSERT INTO Bestand VALUES ('Kosmetik', 1744, 11);
DELETE FROM Artikel WHERE Artikelnr = 1401;
UPDATE Bestand SET Abteilungsname = 'Kleidung'
    WHERE Abteilungsname = 'Textilien';
```

d)

```
/*! Reparieren von insert into Bestand values
('Kosmetik', 1744, 11); */
INSERT INTO Artikel VALUES
  (1744, 'Zahnpasta', 1.99, 1.00, NULL);
/*! Reparieren von delete from Artikel
where Artikelnr = 1401; */
DELETE FROM Bestand WHERE Artikelnr = 1401;
/*! Reparieren von update Bestand set Abteilungsname = 'Kleidung'
where Abteilungsname = 'Textilien';*/
UPDATE Abteilungen SET Abteilungsname = 'Kleidung'
    WHERE Abteilungen = 'Textilien';
```

Aufgabe 2

Funktionale Abhängigkeit meint, dass ein Attribut zu einem anderen Attribut in der gleichen Relation eine Beziehung hat, z.B. wenn ich die Nummer des Bauteils kenne, kenne ich dann die Beschreibung des Bauteils? Volle funktionale Abhängigkeit meint, dass auf der linken Seite der Abhängigkeit nur die notwendingen Attribute stehen.
Bauteil:

- Nr ⇒ BBez
- Nr ⇒ BVorrat
- Nr ⇒ Hersteller
- Nr ⇒ EP

Gerät:

- GBez ⇒ VP
- GBez ⇒ GVorrat
- GBez ⇒ freie_Lieferung
- VP ⇒ freie_Lieferung

Kunde:

- Name, Adr ⇒ Branche
- Adr ⇒ Betreuer

wird_benötigt_für:

- -

wird_geliefert_an:

- GBez, Name, Adr ⇒ Sonderkonditionen

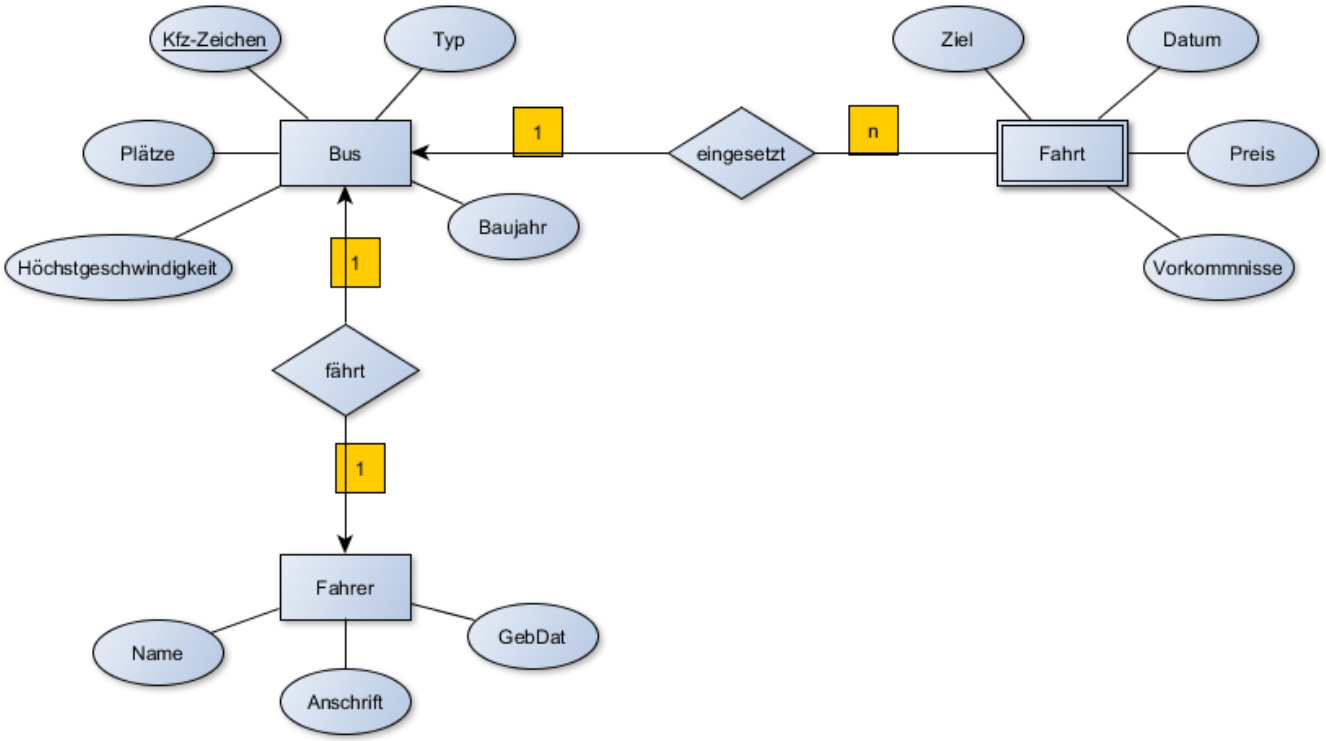
Schlüsselkandidaten

Bauteil: {Nr}
Gerät: {GBez}
Kunde: {Name, Adr}
wird_benötigt_für: {Nr, GBez}
wird_geliefert_an: {GBez, Name, Adr}

Übung 9

Aufgabe 1

a)



Datum ist Diskriminator
b) Relationship-Typen, die starken mit schwachen verbinden, werden nicht konvertiert, es sei denn, sie haben eigene Attribute.

- Bus (Kfz-Zeichen, Typ, Plätze, Baujahr, Höchstgeschw.)
- Fahrer (Name, Anschrift, GebDat)
- Fahrt (Kfz-Zeichen, Datum, Ziel, Preis, Vorkommnisse)
- fährt (Kfz-Zeichen, Name)

c) Bus:

- Kfz-Zeichen ⇒ Typ
- Kfz-Zeichen ⇒ Plätze
- Kfz-Zeichen ⇒ Baujahr {Kfz-Zeichen}
- Kfz-Zeichen ⇒ Höchstgeschw
- Typ ⇒ Plätze

Fahrer:

- Name ⇒ Anschrift {Name}
- Name ⇒ GebDat

Fahrt:

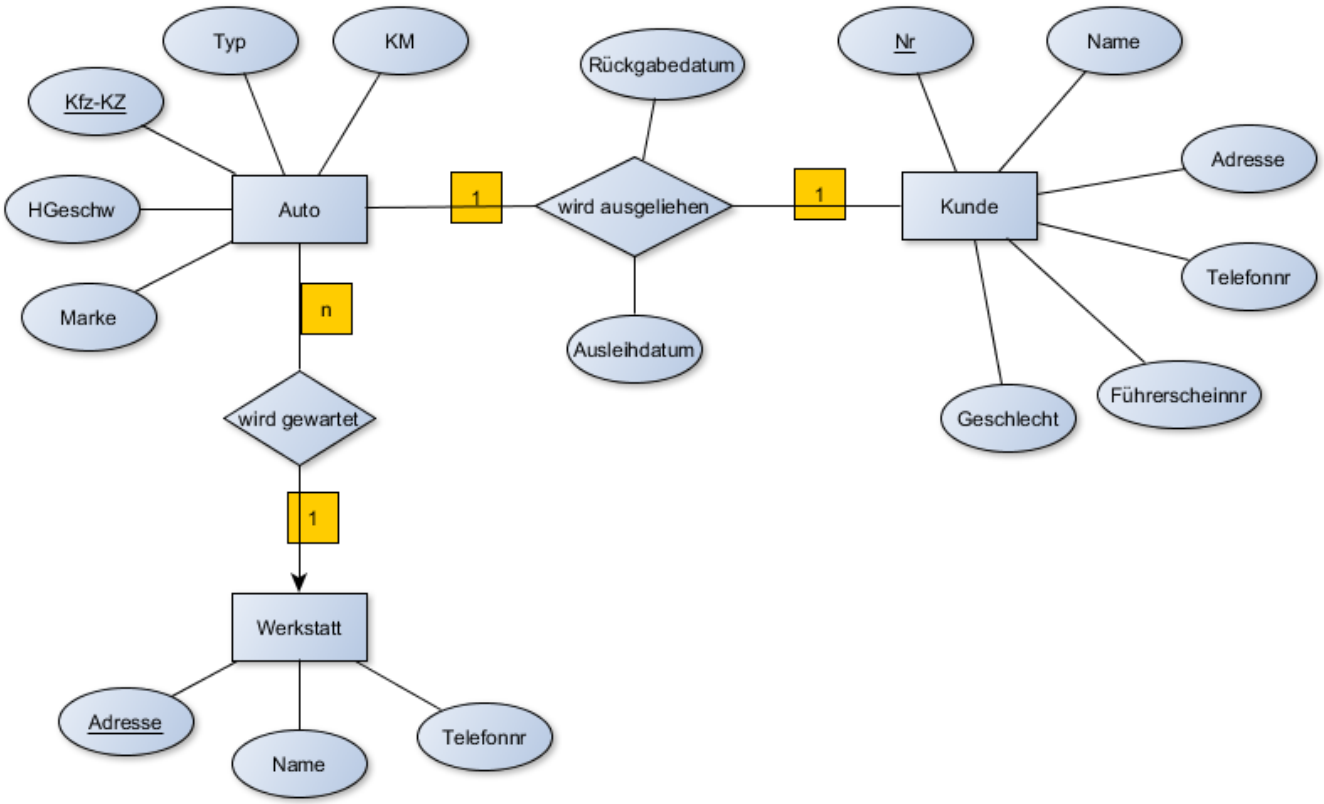
- Kfz-Zeichen, Datum ⇒ Ziel {Kfz-Zeichen, Datum}
- Kfz-Zeichen, Datum ⇒ Preis
- Kfz-Zeichen, Datum ⇒ Vorkommnisse

fährt:

- Kfz-Zeichen ⇒ Name {Kfz-Zeichen}, {Name}
- Name ⇒ Kfz-Zeichen

Aufgabe 2

a)



b)

- Auto(Kfz-KZ, Marke, HGeschw, Typ, KM)
- Kunde(Führerscheinnr, Geschlecht, Nr, Name, Adresse, Telefonnr)
- Werkstatt(Adresse, Name, Telefonnr)
- wird_ausgeliehen(Rückgabedatum, Ausleihdatum, Kfz-KZ, Führerscheinnr)
- wird_gewartet(KFz-KZ, Adresse)

c) Auto:

- Kfz-KZ ⇒ Marke
- Kfz-KZ ⇒ HGewschw
- Kfz-KZ ⇒ Typ
- Kfz-KZ ⇒ KM
- Typ ⇒ Marke

Kunde:

- Führerscheinnr ⇒ Geschlecht
- Führerscheinnr ⇒ Nr
- Führerscheinnr ⇒ Name
- Führerscheinnr ⇒ Adresse
- Führerscheinnr ⇒ Telefonnr
- Nr ⇒ Geschlecht
- Nr ⇒ Name
- Nr ⇒ Adresse

- Nr ⇒ Telefonnr
- Nr ⇒ Führerscheinnr
- Name ⇒ Geschlecht
- Name, Adresse ⇒ Nr
- Name, Adresse = Führerscheinnr
- Name, Adresse ⇒ Telefonnr
- Telefonnr ⇒ Adresse
- Name, Telefonnr ⇒ Nr
- Name, Telefonnr ⇒ Führerscheinnr

Werkstatt:

- Adresse ⇒ Name
- Adresse ⇒ Telefonnr
- Telefonnr ⇒ Adresse
- Telefonnr ⇒ Name

wird_ausgeliehen:

- Kfz-KZ, Ausleihdatum ⇒ Kundennr
- Kfz-KZ, Ausleihdatum ⇒ Rückgabedatum
- Kfz-KZ ⇒ Führerscheinnr
- Kundennr, Ausleihdatum ⇒ Kfz-KZ
- Kundennr, Ausleihdatum ⇒ Rückgabedatum

wird_gewartet:

- Kfz-KZ ⇒ Adresse

Schlüssel:

- Auto: {Kfz-KZ}
- Kunde: {Führerscheinr}, {Nr}, {Name, Adresse}, {Name, Telefonnr}
- Werkstatt: {Adresse}, {Telefonnr}
- wird_ausgeliehen: {Kfz-KZ, Ausleihdatum}, {Kundennr, Ausleihdatum}
- wird_gewartet: {Kfz-Zeichen}

Übung 10

Aufgabe 1

a) 1. NF: nur atomare Werte erlaubt.
2. NF: Jedes Nicht-Schlüsselattribut ist von jedem Schlüsselkandidaten voll funktional abhängig. Bei einem Schlüsselkandidaten muss die Relation in 2.NF sein.
3. NF: Jedes Nicht-Schlüsselattribut darf nur von einem Schlüsselkandidaten voll funktional abhängig sein.
Boyce Codd: Jedes Attribut darf nur von Schlüsselkandidaten abhängen. Auf der linken Seite dürfen nur Schlüsselkandidaten stehen. Blatt 8:

- 1. NF: ja bei allen.

- 2. NF: Kunde nicht ⇒ Relation wird aufgespalten, d.h. das Attribut, das den Verstoß verursacht hat (das rechte), fliegt raus und erhält das Attribut der linken Seite mit dazu. ⇒ Kunde1(Name, Adr, Branche), Kunde2(Adr, Betreuer)
- 3. NF: Gerät nicht. ⇒ Relation wird aufgespalten, d.h. das Attribut, das den Verstoß verursacht hat (das rechte), fliegt raus und erhält das Attribut der linken Seite mit dazu. ⇒ Gerät1(GBez, VP, GVorrat), Gerät2(VP, freie Lieferung)

Blatt 8, 2:

- 1. NF: ja.
- 2. NF: Buch nein ⇒ Buch1(Bestnr, Titel, Autor, Jahr, Signatur), Buch2(Gattung, Titel)
- 3. NF: Entleiher nein ⇒ Entleiher1(BenNr, Name, Adr), Entleiher2(Adr, auswärtig)

Bus, Blatt 9:

- 1. NF: ja.
- 2. NF: ja.
- 3. NF: Bus nicht ⇒ Bus1(Kfz-Zeichen, Typ, Baujahr, Höchstgeschw), Bus2(Typ, Plätze)

b) Ja alle bei Bücherei.

Aufgabe 2

```
UPDATE Gerät SET VP = VP-60 WHERE GBez = 'Drucker';
UPDATE Kunde SET Adr = 'Regensburg'
  WHERE (name='Müller') AND (Adr='Hamburg');
```

Bei 1 wird falsch geupdated, da freie_Lieferung auf false abgeändert werden müsste. ⇒ Gerät1(GBez, VP, GVorrat) und Gerät2(VP, freie_Lieferung).

```
UPDATE Gerät1 SET VP=VP-60 WHERE GBez='Drucker';
```

Aber: Wozu Tabelle Gerät2? Was soll dort gespeichert werden? D.h. Spaltung wird rückgängig gemacht und ein Check eingebaut werden.

```
ALTER TABLE Gerät ADD CHECK ((VP>=500)=freie_Lieferung);
/* Eigentlich mit Transaktion zu lösen => zwei Update Befehle */
```

Bei 2 ist nun ein anderer Betreuer zuständig. ⇒ Kunde1(Name, Adr, Branche) und Kunde2(Adr, Betreuer).

```
UPDATE Kunde1 SET Adr='Regensburg'
  WHERE Name='Müller' AND Adr='Hamburg';
SELECT Betreuer FROM Kunde1, Kunde2
  WHERE Kunde1.Adr= Kunde2.Adr
  AND Name='Müller' AND Kunde1.Adr='Regensburg';
```

Nicht denormalisieren, da in Tabelle nur aufgelistet wird, wo welcher Betreuer zuständig ist.