Relationale Algebra

σ – Selektion (Ausgabe von Zeilen einer Relation, wenn Bedienung erfüllt)

- Bedingungen mit Ausdruck wie; { = , > , < , ≤ , ≥ , ≠ }
- Bsp: $\sigma_{Länge \geq 100}(Filme)$

π – Projektion (Ausgabe von Spalten von einer Relation, Duplikate werden autom. Gelöscht)

- Bsp: $\pi_{Titel,Jahr,Länge}(Filme)$

ρ – Umbenennung

- $\rho_{neuerName}(R)$ → die Relation R erhält somit den Namen „neuerName“
- $\rho_{R(x,y,z)}(R(A,B,C)$ → die Attribute (Spalten) der Relation R haben jetzt die Namen A, B & C

x – Kreuzprodukt (Jede Zeile von Relation 1 wird mit jeder Zeile von Relation 2 kombiniert)

- Zeilen = Anzahl Zeilen R * Anzahl Zeilen S | Anzahl Spalten = Spalten R + Spalten S

R x S

R

A	B
1	2
3	4

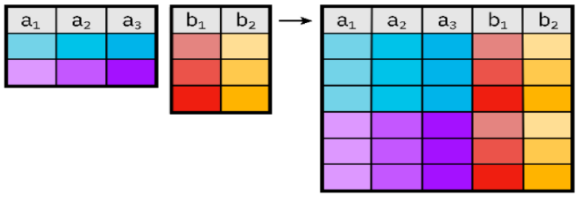
S

B	C	D
2	5	6
4	7	8
9	10	11

R × S

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

a x b



⋈ - Natural Join

- alle Tupel werden miteinander kombiniert, deren gemeinsamen Attribute (Spalten) übereinstimmen. Duplikate werden gelöscht
- Gibt es **keine** gemeinsamen Attribute → **Kreuzprodukt**
- R ⋈ S

R

A	B	C
1	2	3
6	7	8
9	7	8

S

B	C	D
2	5	6
2	3	5
7	8	10

R ⋈ S

A	B	C	D
1	2	3	5
6	7	8	10
9	7	8	10

$\bowtie_P$ - Theta-Join

- bildet Kreuzprodukt mit Join-Bedingung und löscht alle Tupel die Bedingung nicht erfüllen
- P: Join-Bedingung mit logischem Ausdruck { = , > , < , ≤ , ≥ , ≠ }
- Alle Attribute (Spalten) werden zusammengesetzt
- R ⋈p S = σp (R x S)

R

A	B	C
1	2	3
6	7	8
9	7	8

S

B	C	D
2	5	6
2	3	5
7	8	10

R $\bowtie_{A<D}$ S

A	R.B	R.C	S.B	S.C	D
1	2	3	2	5	6
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

R $\bowtie_{A<D \text{ AND } R.B \neq S.B}$ S

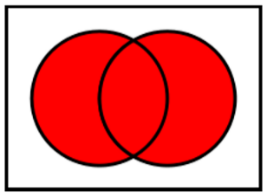
A	R.B	R.C	S.B	S.C	D
1	2	3	7	8	10

Mengenoperatoren

Für Mengenoperationen auf Relationen müssen die **Relationen dasselbe Schema** (gleiche Attribute) und denselben Wertebereich aufweisen. Es gibt in der Mengensemantik **keine Duplikate**.

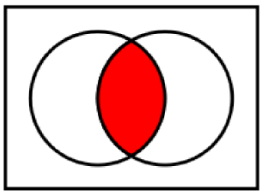
∪ - Vereinigung (Union)

R ∪ S = R'



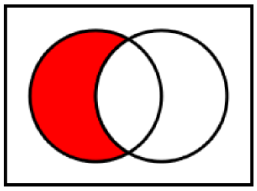
∩ - Schnittmenge

R ∩ S = R'



\ - Differenz

R \ S = R'



Outer Joins

Natural Join (verbindet nur Tupel, bei welcher der Wert in beiden Relationen gleich vorkommt)

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⋈

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1

Left outer Join (alle Tupel der linken Relation wird angezeigt, von der rechten werden nur Tupel übernommen, mit welcher der Wert mit der linken Relation übereinstimmt. Restliche Daten werden mit «NULL» bezeichnet = nicht definiert/ unbekannt)

Bsp: *city* ⟕ *city.name=weather.city* *weather*

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⋈

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	NULL	NULL

Right outer Join (umgekehrt von Left outer Join, nun behalten wir die Rechte Relation)

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⋈

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
NULL	NULL	c_3	d_2	e_2

Full outer Join (alle Tupel werden angezeigt, für Tupel mit einem Match in beiden Relationen werden die Werte komplett angezeigt, ansonsten jeweils die Tupel separat von der linken und rechten mit «NULL» gefüllt

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⋈

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

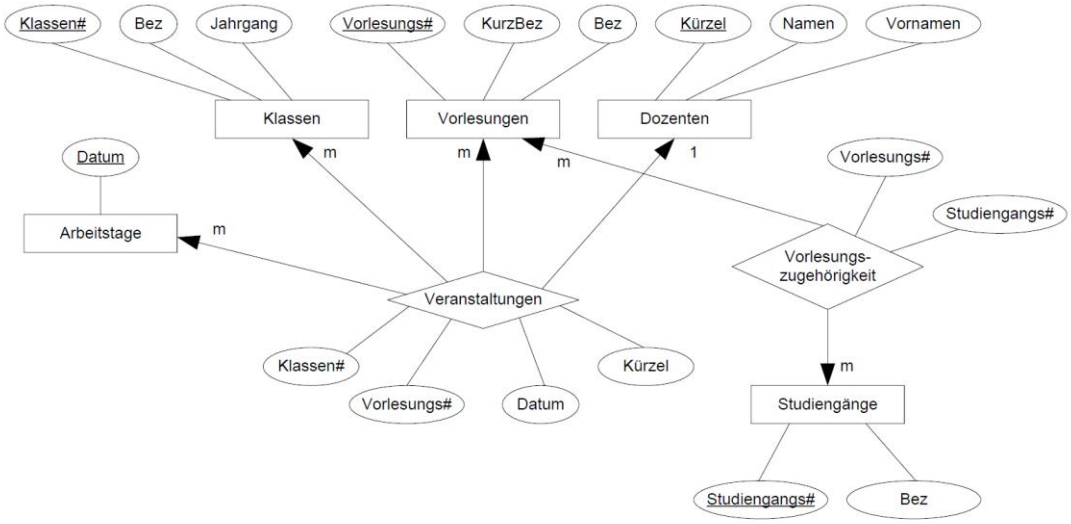
Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	NULL	NULL
NULL	NULL	c_3	d_2	e_2

Bag Algebra

Grösster Unterschied zwischen relationalen Algebra und Bag Algebra = Duplikate werden als normale Tupel beachtet und die Multiplizität (Anzahl) von Duplikaten spielt eine Rolle. Für die Anzahl Duplikate zählt man die rechte Relation und die linke.

Operation	*Anpassung von relationaler Algebra*
Selektion σ	Gleich, Duplikate wie andere Tupel
Projektion π	Gleich, keine Duplikatsentfernung
Kreuzprodukt $\times$	Gleich, Duplikate wie andere Tupel
Joins $\rhd\lhd$	Gleich, Duplikate wie andere Tupel
Vereinigung (bag union) $\cup$	Gleich, **größere** Anzahl an Duplikaten für die vereinigte Relation
Vereinigung (bag concatenation) $\sqcup$	**Neu**, die Summe Anzahl an Duplikaten für die vereinigte Relation
Durchschnitt (bag intersection)	Gleich, bei Duplikaten die kleinere Anzahl nehmen
**Differenz **	Gleich, bei Duplikaten; Differenz von der Anzahl Duplikate der 1. Relation mit der 2. Relation. Wenn ein Rest vorhanden, übernehmen wir und ansonsten ist 0 die Multiplizität
Duplikat-elimination δ	Entfernt Duplikate, also macht aus einem Bag eine Relation. Syntax: $\delta(R)$

Übungsaufgaben mit Lösungen



Aufgabe 2: Relationale Algebra

Gesucht sind Ausdrücke der relationalen Algebra für:

a. Eine Liste der Klassen (Bezeichnung) und der Dozenten (Name und Vorname), welche diese Klassen unterrichten.
b. Eine Liste der Dozenten (Name und Vorname), welche in Klasse 'IT21t' Vorlesungen halten, aber keine Vorlesungen in 'IT21a' halten.
c. Eine Liste der Dozenten (Name und Vorname), welche noch nie eine Vorlesung gehalten haben.
d. Eine Liste der Vorlesungen (Kurzbezeichnung und Bezeichnung) welche allen Studiengängen zugeteilt ist.

Lösungen Aufgabe 2

a. $\pi_{Bez,\ Name,\ Vorname}(Klassen \bowtie Veranstaltungen \bowtie Dozenten)$
b. $\pi_{Name,\ Vorname}[(\pi_{Kürzel}\{\sigma_{Bez='IT21t'}(Klassen) \bowtie Veranstaltungen\} / \pi_{Kürzel}\{\sigma_{Bez='IT21a'}(Klassen) \bowtie Veranstaltungen\}) \bowtie Dozent]$
c. $\pi_{Name,\ Vorname}[\{\pi_{Kürzel}(Dozenten) / \pi_{Kürzel}\ (Veranstaltungen)\} \bowtie Dozenten]$
d. $\pi_{Kurzbez,\ Bez}\ [(\pi_{VorlesungsNr}(Vorlesung) / \pi_{VorlesungsNr}\{\pi_{VorlesungsNr,\ StudiengangsNr}(Studiengänge \bowtie Vorlesungen) / \pi_{VorlesungsNr,\ StudiengangsNr}(Vorlesungszugehörigkeit)\}) \bowtie Vorlesungen]$

Einfacher geht es mit der Division, diese wird in der Vorlesung aber nicht behandelt:
$\pi_{Kurzbez,\ Bez}[\{\pi_{VorlesungsNr,\ StudiengangsNr}(Vorlesungszugehörigkeit) \div \pi_{StudiengangsNr}(Studiengänge)\} \bowtie Vorlesungen]$

Aufgabe 3: Relationale Bag-Algebra

Gesucht sind Ausdrücke der relationalen Bag-Algebra für:

a. Eine Liste der Klassen (Bezeichnung) welche zwischen dem 1.1.2021 und dem 31.12.2021 Vorlesungen hatten.
b. Eine Liste der Dozenten (Name und Vorname) welche im Studiengang 'Bachelorstudiengang Informatik' unterrichten.
c. Eine Liste der Studiengänge (Bezeichnung), bei welchen Dozent Müller Hans keine Vorlesungen hält.
d. Eine Liste der Veranstaltungen (KurzBez der Vorlesung, Name des Dozenten und Datum), welche als KurzBez 'DAB1' haben oder deren Dozent den Namen 'Müller' hat. Eine Veranstaltung, die beide Bedingungen erfüllt soll **doppelt** in der Liste erscheinen ($\sqcup$ oder $\cup$ zur Bildung des Resultats verwenden).

Lösungen Aufgabe 3

a. $\delta[\pi_{Bez}(\sigma_{Datum>='1.1.2021'\ AND\ Datum<='31.12.2021'}(Veranstaltungen \bowtie Klassen)]$
b. $\delta[\pi_{Name,\ Vorname}\{Veranstaltungen \bowtie Vorlesungszugehörigkeit \bowtie \sigma_{Studiengänge.Bez='Bachelorstudiengang\ Informatik'}(Studiengänge) \bowtie Dozenten\}]$
c. $\pi_{Studiengänge.Bez}$
$([\pi_{StudiengangsNr}(Studiengänge) \setminus \pi_{StudiengangsNr}\{\sigma_{Name='Müller'\ AND\ Vorname='Hans'}(Vorlesungszugehörigkeit \bowtie Veranstaltungen \bowtie Dozenten)\}] \bowtie Studiengänge)$
d. $(\delta[\pi_{KurzBez,\ Name,\ Datum}\{\sigma_{Vorlesung.KurzBez='DAB1'}(Vorlesung \bowtie Veranstaltung \bowtie Dozenten)\}]) \cup$
$(\delta[\pi_{KurzBez,\ Name,\ Datum}\{\sigma_{Vorlesung.Name='Müller'}(Vorlesung \bowtie Veranstaltung \bowtie Dozenten)\}])$

Weitere Aufgaben zu rel. Algebra

Aufgabe 2

Gegeben sind die Relationenformate:

Gast(Besucher, Restaurant)

Sortiment(Restaurant, Biersorte)

Vorzug(Besucher, Biersorte)

sowie je eine Relation, *g* zum Format *Gast*, *s* zum Format *Sortiment*, und *v* zum Format *Vorzug*.

a) Alle Biersorten, die Besucher Müller vorzieht
b) Alle Restaurants, welche die Biersorte Cardinal nicht im Sortiment haben
c) $(g \bowtie s) \setminus (g \bowtie s \bowtie v)$
d) $\left(\pi_{Besucher}(g) \cup \pi_{Besucher}(v)\right) \bowtie \left(\pi_{Restaurant}(g) \cup \pi_{Restaurant}(s)\right) \bowtie \left(\pi_{Biersorte}(v) \cup \pi_{Biersorte}(s)\right)$
e) Alle Restaurants, die Gäste haben und Bier im Sortiment, aber keinen Gast der irgendeines seiner Vorzugsbiere erhalten könnte
f) Die Restaurants mit Bieren im Sortiment, die nie verlangt werden
g) $\left(\pi_{Restaurant}(g) \setminus \pi_{Restaurant}(s)\right) \cup \left(\pi_{Restaurant}(s) \setminus \pi_{Restaurant}(g)\right)$
h) Die Restaurants, welche keine Gäste haben
i) Die Restaurants, die zwar Gäste haben, aber keine Bier trinkenden
j) Die Restaurants mit lauter Bier trinkenden Gästen, welche keines ihrer Vorzugsbiere erhalten können

Lösungen:

a) $\pi_{Biersorte}\left(\sigma_{Besucher='Müller'}(v)\right)$
b) $\left(\pi_{Restaurant}(s) \cup \pi_{Restaurant}(g)\right) \setminus \pi_{Restaurant}\left(\sigma_{Biersorte='Cardinal'}(s)\right)$
Die erste Hälfte ist wichtig, da ALLE Restaurants gefragt sind, auch solche ohne Bier im Sortiment
c) Alle Tupel <x, y, z>, so dass Besucher x im Restaurant y Gast ist und y die Biersorte z im Sortiment hat und z nicht zu den Vorzugsbiersorten von x gehört.
d) Alle Kombinationen <x, y, z> aller vorkommenden Besucher x und Restaurants y und Biersorten z.
e) $\left(\pi_{Restaurant}(g) \cap \pi_{Restaurant}(s)\right) \setminus \pi_{Restaurant}(g \bowtie s \bowtie v)$
f) $\pi_{Restaurant}\left(\left(\pi_{Biersorte}(s) \setminus \pi_{Biersorte}(v)\right) \bowtie s\right)$
g) Die Restaurants, welche keine Gäste haben oder kein Bier
h) $\pi_{Restaurant}(s) \setminus \pi_{Restaurant}(g)$
i) $\pi_{Restaurant}(g) \setminus \pi_{Restaurant}(g \bowtie v)$
j) $\pi_{Restaurant}(g \bowtie v) \setminus \pi_{Restaurant}(g \bowtie v \bowtie s)$

Übungsaufgabe Bag-Algebra

e) Im Folgenden gelten die Gesetze der **Bagalgebra** (**NICHT** relationale Algebra). Gegeben seien die zwei Formate $F_1(A,B,C)$ und $F_2(B,C,D)$. Alle Domänen seien die Menge {0,1}, also booleans, sowie die **Bags** x zu F_1 und y und z zu F_2:

x	A	B	C
	1	1	0
	1	0	1
	1	1	0

y	B	C	D
	0	1	0
	0	1	1
	1	1	1

z	B	C	D
	0	1	1
	0	1	0
	1	1	0

Berechnen Sie folgenden Ausdruck:

$$r' = \left(\pi_{C,B}(z) \sqcup \pi_{C,B}(y \bowtie x)\right)$$

Stellen Sie das Resultat **als Tabelle** dar.

1) Zuerst bei der inneren Klammer starten (*Join von x und y*) = welche Werte B & C sind gleich in beiden Tabellen und diese selektieren

x	A	B	C
	1	1	0
	1	0	1
	1	1	0

y	B	C	D
	0	1	0
	0	1	1
	1	1	1

2) Vereinigung mit (z) untereinander in der Reihenfolge der Aufgaben (z) dann (y & x)

Schritt 4: Vereinigung ($\pi_{C,B}(z) \cup \pi_{C,B}(y \bowtie x)$):

$\pi_{C,B}(z)$:

$$\pi_{C,B}(z) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Vereinigung mit $\pi_{C,B}(y \bowtie x)$:

$$r' = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Entity Relationship

ER Begriffe & Darstellung

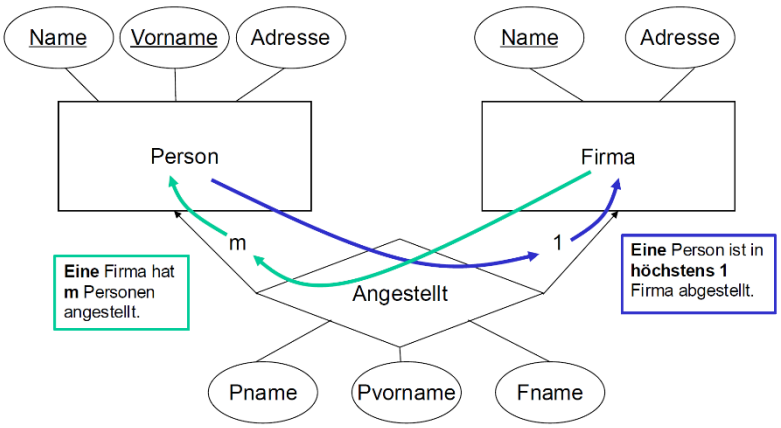
Entitätstyp (Tabelle) = Rechteck

Attribute = Ovale

Primärschlüssel werden unterstrichen

Beziehungstypen = Rhombus

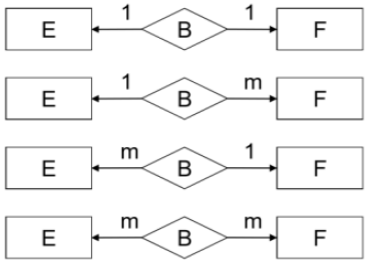
Pfeile für Verbindungen mit 1 oder m



Primärschlüssel
existieren nur für Entitätstypen, auf welche die Pfeile draufzeigen

Schlüssel und Beziehungen

Schlüssel werden in {} notiert und es gibt zwei Beziehungstypen = **1 oder M**



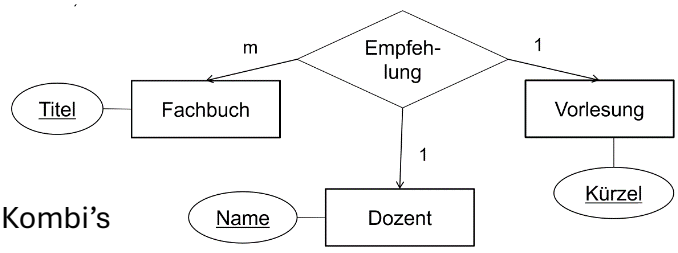
- Bei einer 1:M Beziehung liegt der Schlüssel immer bei dem Entitätstypen mit M, da der andere Typ durch die zu 1 Beziehung eindeutig zugewiesen werden kann.
- Bei einer 1:1 Beziehung muss es von jedem Entitätstypen einen Schlüssel geben.
- Bei einer M:M Beziehung muss ein kombinierter Schlüssel aus beiden erzeugt werden

Sonderfälle

M:M:M → Einen zusammengesetzten Schlüssel aus **drei** Attributen

1:M:M → Einen zusammengesetzten Schlüssel aus **zwei** Attributen der **M**s

1:1:M → **2** Schlüssel; der beiden **1** Beziehungen



- **für Vorlesung {Titel, Name}**
- **für Dozent {Titel, Kürzel}**

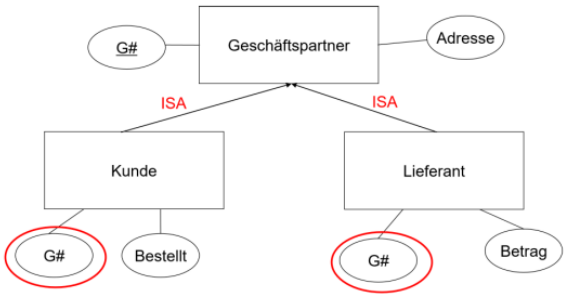
1:1:1 → **3** Zusammengesetzte Schlüssel, mit allen Kombi's

- {A1; A2}, {A2; A3}, {A1; A3}

Abhängigkeiten

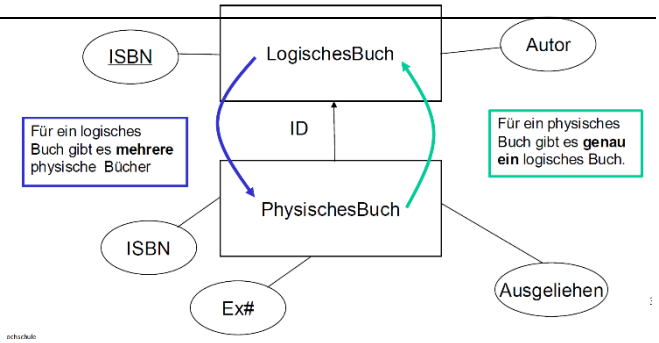
ISA - ABHÄNGIGER ENTITÄTSTYP

- erben alle Attribute von den Parent-Entitäten
- Übernehmen Primärschlüssel von Parent (Geschäftspartner)
- ist eine abhängige 1:1 Beziehung



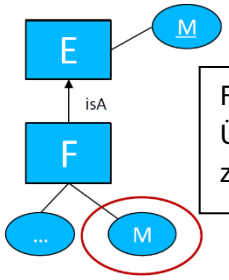
ID - ABHÄNGIGER ENTITÄTSTYP

- Existiert nur mit Parent – Entitäten
- Übernimmt Schlüssel von Parent (ISBN) + zusätzlichen Schlüssel (Ex#)
- Nur Primärschlüssel, wenn Pfeil drauf zeigt Bsp. = Bestellpos., Lieferpos. Etc.



ISA-abhängig

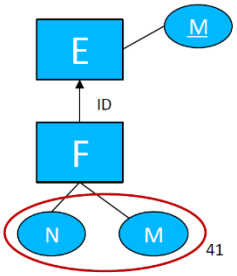
- Genauere Spezifikation → Spezialisierung
- Ist Entitätstyp F von Entitätstyp E ISA-abhängig, so gilt:
  - Ist M die Menge der Primärschlüsselattribute in E, so muss M in F ein Schlüssel sein.



Für Entitäten, welche eine Überkategorie sind wie Mitarbeiter zu Manager/ Verkäufer

ID-abhängig

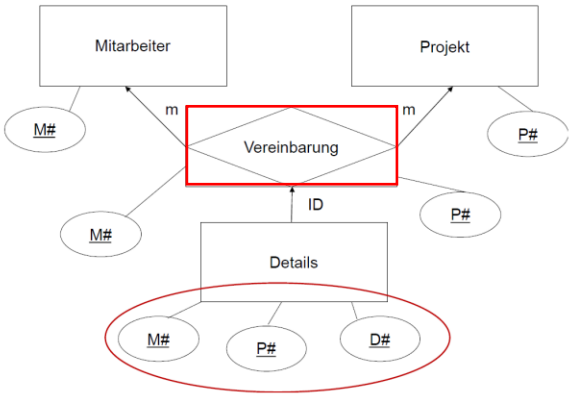
- Hierarchie, hängt an "Oberklasse". Ist nur innerhalb der Hierarchie definiert
- Ist Entitätstyp F von Entitätstyp E ID-abhängig, so gilt:
  - Ist M die Menge der Primärschlüsselattribute in E, so muss M ∪ N ein Schlüssel in F sein, wobei N eine zu M elementfremde Menge von Attributen von F ist (min. 1 Element)



41

Zusammengesetzter Entitätstyp

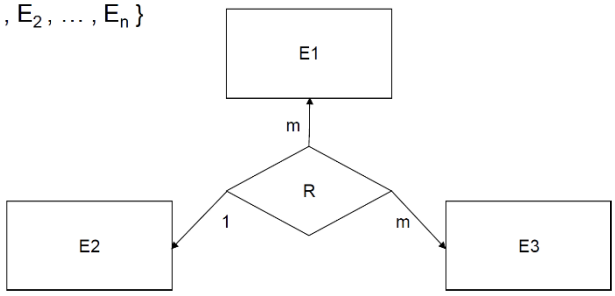
Es ist mindestens ein weiteres Attribut nötig. So kann z.B. {M#,P#,D#} ein Schlüssel sein (wegen ID-Abhängigkeit)



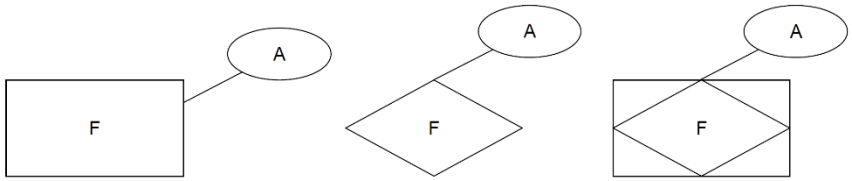
Erstellung von ER-Modell

1. **Definiere unabhängigen Entitätstyp**
   Darstellung: ein neues Rechteck (mit eindeutigem Namen)

2. **Definiere Beziehungstyp**
   Darstellung: Rhombus mit Beziehungstyp 1 oder M mit Pfeilen auf die Entitätstypen gerichtet
   $\{E_1, E_2, \dots, E_n\}$



3. **Definiere Attribut**
   Darstellung: Ovale strichverbunden zu allen Entitätstypen mit eindeutigem Namen



4. **Umwandlung Beziehungstyp in zusammengesetzten Entitätstyp**
   Darstellung: Rhombus im Rechteck umschlossen nur für ID-Abhängige Entitäten

5. **Definiere ID & ISA-abhängigen Entitätstyp**
   Darstellung: Pfeile mit ID/ ISA auf Entitätstypen gerichtet
   ID – Beispiele: Lieferposition <- Lieferung, Projektaufgaben zu Projekten

   ISA für Überkategorie zu Unterkategorie; Mitarbeiter zu Leiter, Verkäufer, Service/ Fahrzeuge zu Autos, Motorrad, Velo

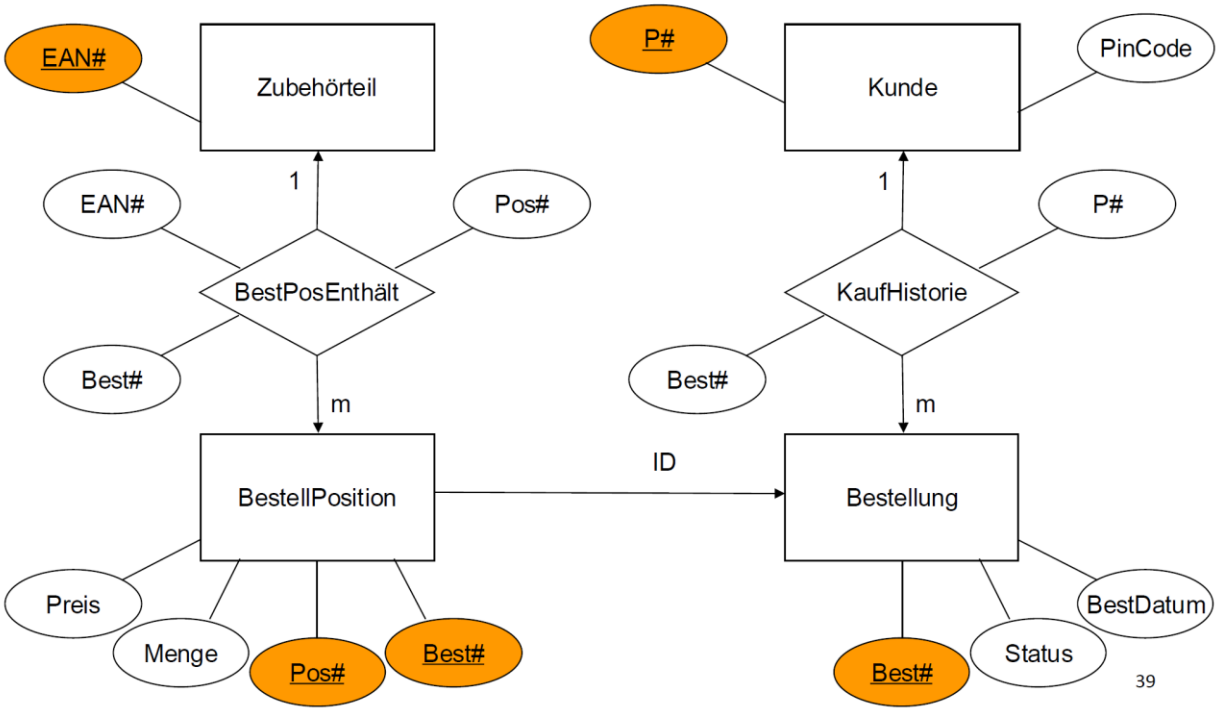
6. **Schlüssel definieren**

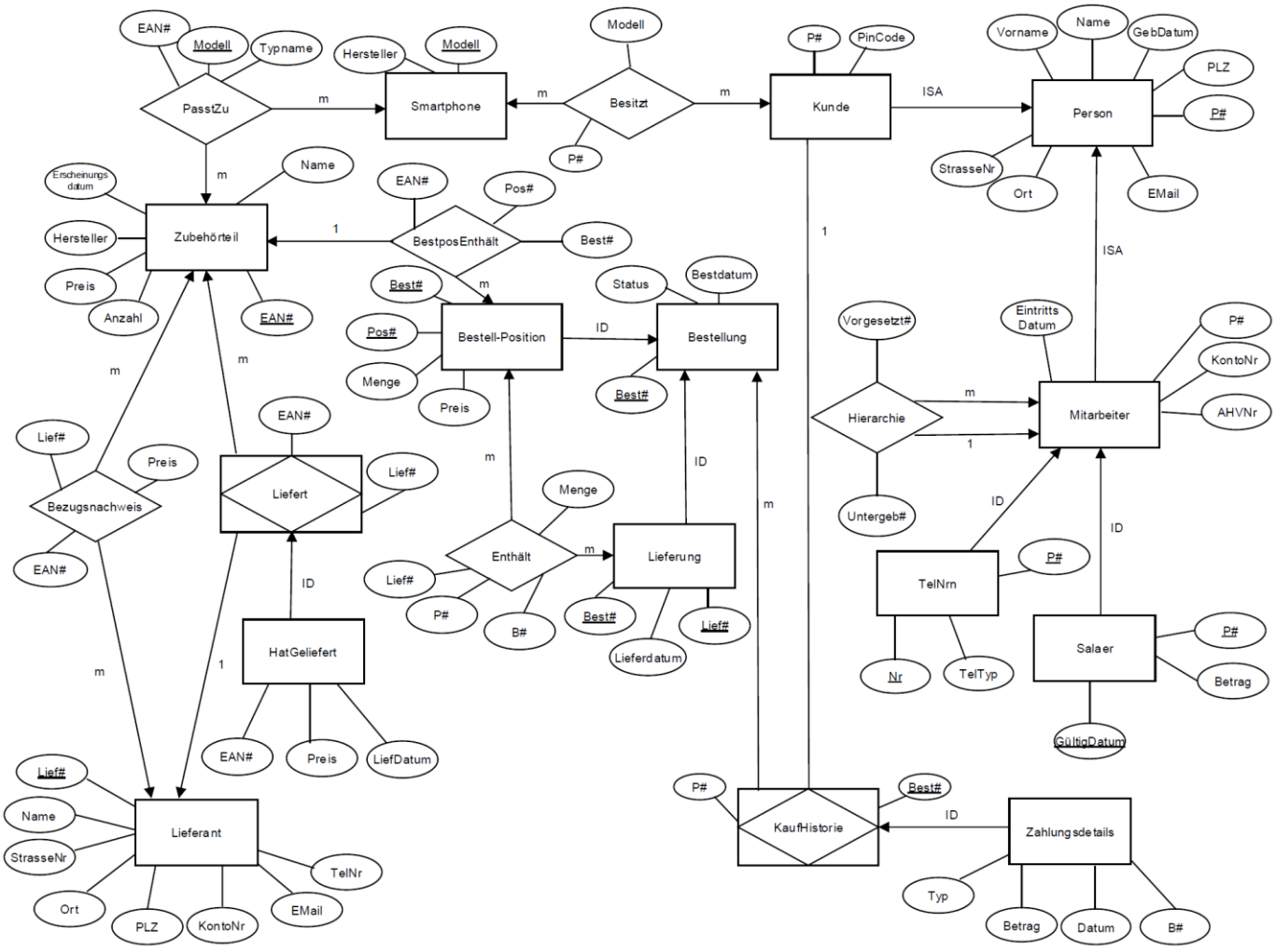
Für das relationale Modell müssen wir noch Primär und Fremdschlüssel definieren. Sowie die Schlüssel (Constraints) die sich aus den Beziehungen ergeben.

Jeder unabhängige Entitätstyp erhält einen oder mehrere Schlüssel:

- Falls der Entitätstyp eingehende Pfeile hat, wählen wir **einen <u>Primärschlüssel</u>**
- Für Beziehungstypen: wir wählen Fremdschlüssel und Schlüssel (Unique Constraints) (gemäss Kardinalitäten = 1 oder m)
- Für Umwandlung in zusammengesetzte Entitätstypen: Primärschlüssel wählen
- Entitätstyp E ist ID- oder ISA-abhängig von F: Primärschlüssel in F wählen, Fremdschlüssel und Schlüssel (Unique Constraints) in E wählen

Beispiele von komplexen ER-Modellen





SQL

Umwandlung relationale Algebra in SQL-Statement

RELATIONALE ALGEBRA → SQL

$\sigma_{a=3}(R)$	SELECT * FROM R WHERE a=3
$\pi_{a,b}(R)$	SELECT a, b FROM R
$\pi_{a\to x,3*b\to y}(R)$	SELECT a AS x, 3*b AS y FROM R
$\rho_{S(d,e)}(R(a,b))$	SELECT a AS d, b AS e FROM R AS S
R x S	SELECT * FROM R,S
	SELECT * FROM R CROSS JOIN S
R ⋈ S	SELECT * FROM R NATURAL JOIN S
$R \bowtie_{R.a <= S.c} S$	SELECT * FROM R, S WHERE R.a <= S.c
	SELECT * FROM R JOIN S ON R.a <= S.c
R ⟕ S	SELECT * FROM R LEFT OUTER JOIN S
R ∩ S	SELECT * FROM R INTERSECT SELECT * FROM S
R ∪ S	SELECT * FROM R UNION SELECT * FROM S
R \ S	SELECT * FROM R EXCEPT SELECT * FROM S
δ(R)	SELECT DISTINCT * FROM R

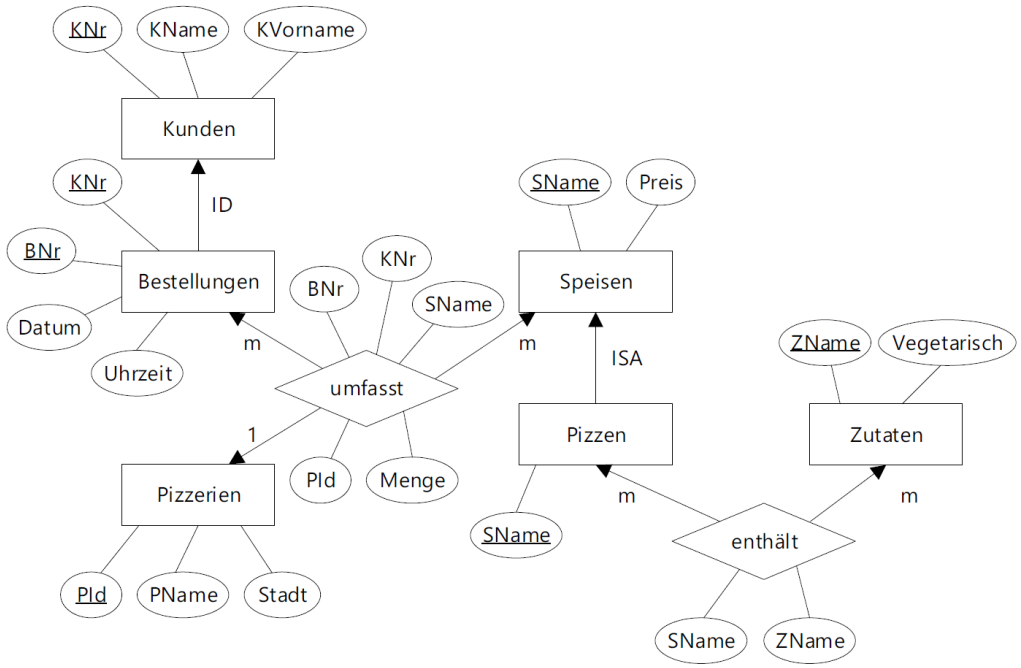
- Vereinigung ⊔ → UNION ALL

$$\pi_{Name,Vorname}(Besucher) \setminus \left(\pi_{Bname,Bvorname}(Gast) \sqcup \pi_{Bname,Bvorname}(Lieblingsbier)\right)$$

In SQL:

```
SELECT Name, Vorname
FROM Besucher
EXCEPT
(SELECT BName, BVorname From Gast
UNION ALL
SELECT BName, BVorname FROM Lieblingsbier);
```

Probeprüfung Aufgaben & Lösungen



a) Schreiben Sie die SQL-Statements für die **Erstellung der Tabellen** "Bestellung" und "umfasst" im angegebenen ER-Diagramm. Ihre Lösung muss alle Schlüssel und ggf. weitere Constraints umfassen. Nehmen Sie an, dass die restlichen Tabellen bereits geeignet angelegt worden sind.

```
CREATE TABLE Bestellungen (
    BNr INT NOT NULL,
    KNr INT NOT NULL,
    PId INT NOT NULL,
    Datum DATE NOT NULL,
    Uhrzeit TIME NOT NULL,
    PRIMARY KEY (BNr),
    CONSTRAINT FK_Bestellungen_Kunden FOREIGN KEY (KNr) REFERENCES Kunden(KNr),
    CONSTRAINT FK_Bestellungen_Pizzerien FOREIGN KEY (PId) REFERENCES
Pizzerien(PId)
);

CREATE TABLE umfasst (
    BNr INT NOT NULL,
    SName VARCHAR(50) NOT NULL,
    Menge INT NOT NULL CHECK (Menge > 0),
    PRIMARY KEY (BNr, SName),
    CONSTRAINT FK_umfasst_Bestellungen FOREIGN KEY (BNr) REFERENCES
Bestellungen(BNr),
    CONSTRAINT FK_umfasst_Speisen FOREIGN KEY (SName) REFERENCES Speisen(SName)
);
```

b) **Fügen** Sie für eine einzelne Bestellung bei derselben Pizzeria zwei Bestellpositionen mit zwei verschiedenen Speisen in die Tabelle "umfasst" **ein**. Gehen Sie davon aus, dass alle benötigten, referenzierten Datensätze bereits existieren:

```
INSERT INTO umfasst (BNr, SName, Menge, Pld)
VALUES     (1, 'Pizza Margherita', 2, 17),
           (1, 'Lasagne', 1, 17);
```

c) Schreiben Sie ein SQL-Statement, welches eine Tabelle mit Kundenname, Kundenvorname, Bestellnummer, Bestellbetrag (**Betrag der gesamten Bestellung**), sowie **Anzahl Positionen** je Bestellung erstellt. Es sollen nur Bestellungen aufgelistete werden, in welchen für mehr als Fr. 70.- bestellt wurde.

```
SELECT
    K.KName AS Kundenname,
    K.KVName AS Kundenvorname,
    B.BNr AS Bestellnummer,
    SUM(S.Preis * U.Menge) AS Bestellbetrag,
    COUNT(U.SName) AS Anzahl_Positionen
FROM Bestellungen B
JOIN Kunden K ON B.KNr = K.KNr
JOIN umfasst U ON B.BNr = U.BNr
JOIN Speisen S ON U.SName = S.SName
GROUP BY K.KName, K.KVName, B.BNr
HAVING SUM(S.Preis * U.Menge) > 70;
```

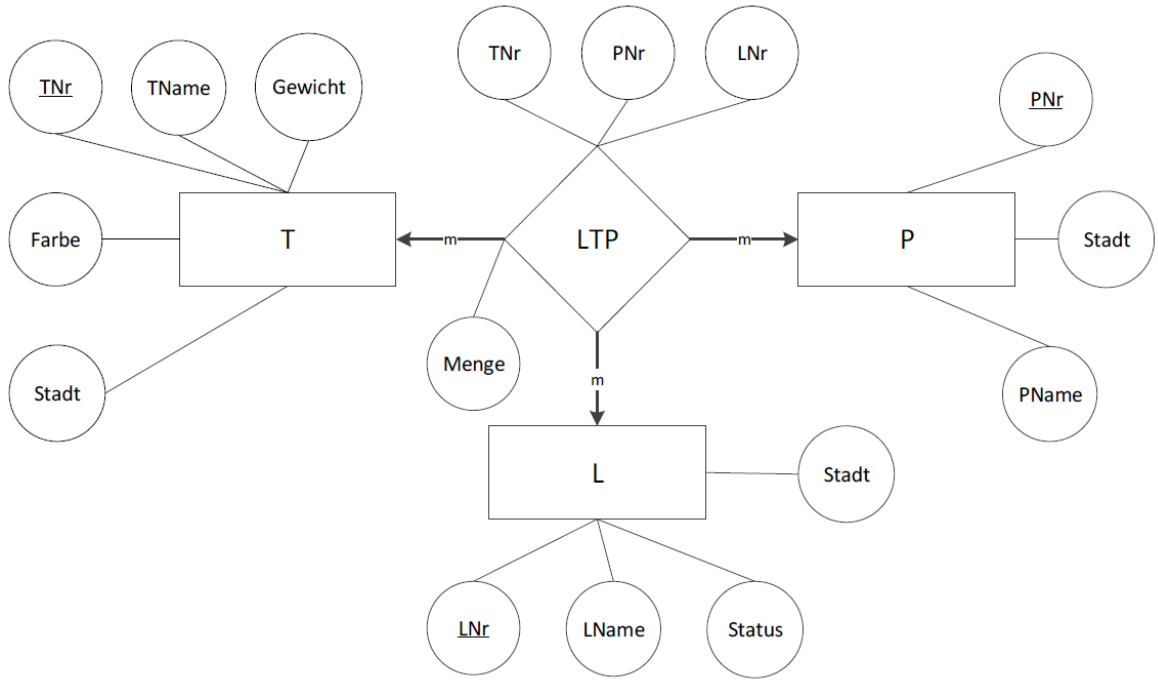
d) Schreiben Sie ein SQL-Statement, welches eine Tabelle mit der **Menge der bestellen Pizzen pro Jahr** erstellt (Menge, Pizzaname und Jahr in der Tabelle auflisten), die **nur vegetarische Zutaten** enthielten.

```
SELECT
    YEAR(B.Datum) AS Jahr,
    P.SName AS Pizzaname,
    SUM(U.Menge) AS Menge
FROM Bestellungen B
JOIN umfasst U ON B.BNr = U.BNr
JOIN Pizzen P ON U.SName = P.SName
JOIN enthält E ON P.SName = E.SName
JOIN Zutaten Z ON E.ZName = Z.ZName
WHERE Z.Vegetarisch = TRUE
GROUP BY YEAR(B.Datum), P.SName;
```

e) Schreiben Sie ein SQL-Statement, welches eine Tabelle mit den Pizzanamen erstellt, welche **alle Zutaten enthalten** und die **höchstens Fr. 20**.- kosten.

```
SELECT P.SName AS Pizzaname
FROM Pizzen P
JOIN enthält E ON P.SName = E.SName
GROUP BY P.SName
HAVING
    COUNT(DISTINCT E.ZName) = (SELECT COUNT(*) FROM Zutaten)
    AND MAX(P.Preis) <= 20;
```

Format	Constraints	Beschreibung
L (LNr, LName, Status, Stadt)	{LNr} Primary Key {LName} Unique	Lieferant
T (TNr, TName, Farbe, Gewicht, Stadt)	{TNr} Primary Key {TName} Unique	Teil
P (PNr, PName, Stadt)	{PNr} Primary Key {PName} Unique	Projekt
LTP (LNr, TNr, PNr, Menge)	{LNr} Foreign Key auf L {TNr} Foreign Key auf T {PNr} Foreign Key auf P {LNr, TNr, PNr} Unique	„Welcher Lieferant liefert welche Teile für welche Projekte in welcher Menge“. Eine Lieferung entspricht einer Zeile in LTP



Aufgaben Level 1

1. Finden Sie alle verschiedenen Teilefarben/Teilestädte-Kombinationen.
2. Finden Sie alle Lieferantennummern/Teilenummern/Projektnummern-Kombinationen, bei denen Lieferant, Teil und Projekt alle aus derselben Stadt kommen.
3. Finden Sie die Teilenummern von allen Teilen, welche von einem Lieferanten aus Winterthur für ein Projekt in Winterthur geliefert werden.
4. Definieren Sie eine Sicht (view) mit Namen BStadtTeile, die alle Angaben über alle Teile liefert die aus einer Stadt stammen deren Namen mit ‚B‘ beginnt. Fragen Sie die Sicht ab nach blauen Teilen (nur die Attribute TName und Gewicht).
5. Finden Sie die Lieferantennummern aller Lieferanten, die ein Teil liefern das aus einer Stadt stammt, deren Namen mit ‚B‘ beginnt (verwenden Sie obige Sicht).
6. Finden Sie die Projektnummern aller Projekte, die mit dem Teil mit der Nummer T1 beliefert werden in einer durchschnittlichen Menge, welche grösser ist als die grösste Menge eines Teiles, das an das Projekt mit der Nummer P1 geliefert wird.
7. Finden Sie die Teilenummern aller Teile, welche an alle Projekte in Winterthur geliefert werden.
8. Finden Sie alle Lieferantennummern mit Status kleiner als der Status von Sulzer.
9. Finden Sie alle Paare von verschiedenen Teilenummern, bei denen es einen Lieferanten gibt, welcher beide Teile liefert.

10. Finden Sie die Anzahl Projekte, zu denen der Lieferant mit dem Namen “Sulzer“ beiträgt.

SQL Statements zu Aufgaben Level 1:

```
1) SELECT DISTINCT Farbe, Stadt FROM T;

2) SELECT L.LNr, T.TNr, P.PNr
FROM (L JOIN T ON L.Stadt = T.Stadt) JOIN P
ON T.Stadt = P.Stadt;

3) SELECT DISTINCT LTP.TNr
FROM LTP, L, P
WHERE LTP.LNr = L.LNr AND LTP.PNr = P.PNr AND
P.Stadt = 'Winterthur' AND L.Stadt = 'Winterthur';

4) CREATE VIEW BStadtTeile AS
SELECT TNr,TName,Farbe,Gewicht,Stadt
FROM T
WHERE Stadt LIKE 'B%';
SELECT TName,Gewicht FROM BStadtTeile WHERE Farbe = 'Blau';

5) SELECT DISTINCT LNr
FROM LTP JOIN BStadtTeile
ON LTP.TNr = BStadtTeile.TNr;

6) SELECT PNr
FROM LTP
WHERE TNr ='T1'
GROUP BY PNr
HAVING AVG(Menge) > (SELECT MAX(Menge)
FROM LTP
WHERE PNr ='P1');

7) SELECT TNr
FROM T
WHERE NOT EXISTS (
SELECT ''
FROM P
WHERE P.Stadt ='Winterthur' AND NOT EXISTS (
SELECT ''
FROM LTP
WHERE LTP.PNr = P.PNr AND LTP.TNr = T.TNr)
);

8) SELECT LNr FROM L
WHERE Status < (SELECT Status FROM L WHERE LName= 'Sulzer');

9) SELECT DISTINCT x.TNr, y.TNr
FROM LTP AS x, LTP AS y
WHERE x.LNr = y.LNr AND x.TNr < y.TNr;

10) SELECT COUNT(DISTINCT LTP.PNr) AS AnzahlProjekte
FROM LTP
JOIN L
ON LTP.LNr = L.LNr
WHERE L.LName = 'Sulzer';
```

Aufgaben Level 2:
1) Finden Sie die Teilenummern von allen Teilen, welche von einem Lieferanten aus derselben Stadt geliefert werden, wie das Projekt, für welches das Teil geliefert wird.
2) Finden Sie die Gesamtmenge von Teil ‚T1', die Sulzer (für irgendein Projekt) liefert.
3) Finden Sie die Teilenummern aller Teile, welche an mindestens ein Projekt mit durchschnittlicher Menge grösser als 350 geliefert werden.
4) Finden Sie die Projektnamen aller Projekte, die durch Sulzer beliefert werden.
5) Finden Sie die Farben aller Teile, die durch Sulzer geliefert werden.
6) Finden Sie alle Teilenummern, welche an irgendein Projekt in Winterthur geliefert werden.
7) Finden Sie die Projektnummern aller Projekte, welche mindestens ein Teil geliefert bekommen, das auch von Sulzer (irgendwohin) geliefert wird.
8) Finden Sie alle Lieferanten/Teile-Paare, bei denen der Lieferant das Teil nicht liefert.
9) Finden Sie die Lieferantennummern aller Lieferanten, welche das Teil mit der Nummer ‚T1' in einer Menge liefern, die grösser ist als die durchschnittliche Menge von Teilen ‚T1', welche an dasselbe Projekt geliefert werden.
10) Finden Sie alle Städte, in denen sich mindestens ein Lieferant, ein Teil oder ein Projekt befindet.

```
1) SELECT DISTINCT LTP.TNr
FROM LTP, L, P
WHERE LTP.LNr = L.LNr AND LTP.PNr = P.PNr AND
L.Stadt = P.Stadt;
```

```
2) SELECT SUM(Menge)
FROM LTP, L
WHERE LTP.LNr = L.LNr AND L.LName ='Sulzer' AND LTP.TNr ='T1';
```

```
3) SELECT DISTINCT TNr
FROM LTP
GROUP BY TNr, PNr
HAVING AVG(Menge) > 350;
```

```
4) SELECT Pname
FROM P
WHERE PNr IN (
SELECT LTP.PNr
FROM LTP, L
WHERE LTP.LNr = L.LNr AND L.LName ='Sulzer'
);
```

```
5) SELECT DISTINCT Farbe
FROM T
WHERE TNr IN (
SELECT LTP.TNr
FROM LTP, L
WHERE LTP.LNr = L.LNr AND L.LName ='Sulzer'
);
```

```
6) SELECT DISTINCT LTP.TNr
FROM LTP, P
WHERE LTP.PNr = P.PNr AND P.Stadt ='Winterthur';
```

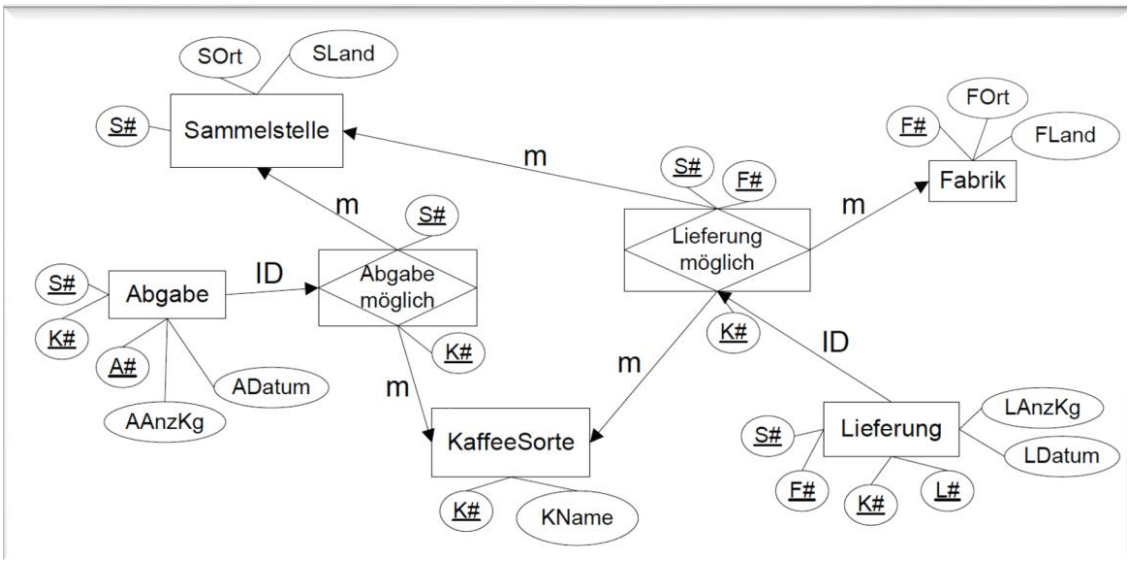
```
7) SELECT DISTINCT PNr
FROM LTP
WHERE TNr IN (
SELECT TNr
FROM LTP, L
WHERE LTP.LNr = L.LNr AND L.LName ='Sulzer'
);

8) (SELECT LNr, TNr FROM L, T)
EXCEPT
(SELECT LNr, TNr FROM LTP);

9) SELECT DISTINCT x.LNr
FROM LTP AS x
WHERE x.TNr ='T1' AND x.Menge > (
SELECT AVG(y.Menge)
FROM LTP AS y
WHERE x.PNr = y.PNr AND y.TNr ='T1'
);

10) SELECT Stadt FROM L
UNION
SELECT Stadt FROM T
UNION
SELECT Stadt FROM P;
```

Bewertetes Praktikum SQL



Aufgabe 2) Erstellen Sie eine Liste von Kaffeesorten (K#, KName), die noch nie an eine Fabrik geliefert worden sind, die aber an eine Sammelstelle abgegeben worden sind.

```
SELECT DISTINCT K.K, kname from kaffeesorte K
JOIN abgabe A ON K.K = A.K
WHERE K.K NOT IN (
SELECT DISTINCT L.K from Lieferung L
);
```

Aufgabe 3) Erstellen Sie eine Liste von Kaffeesorten, die von allen Fabriken geliefert worden sind.

```
SELECT K.K, K.KName
FROM KaffeeSorte K
JOIN Lieferung L ON K.K = L.K --selektion auf alle kaffees die jemals
ausgeliefert wurden
GROUP BY K.K, K.KName --dadurch wird kaffeesorte um kaffeesorte verglichen
HAVING COUNT(DISTINCT L.F) = (SELECT COUNT(*) FROM Fabrik);
-- having ist wie ein select, nur was nachstehend erfüllt wird bleibt in den
ergebnissen
-- count(distinct l.f) -> damit werden die verschiedenen fabriken gezählt (es
könnten ja auch
--eine fabrik mit mehreren lieferungen der gleichen kaffeesorte gegeben haben)
```

Aufgabe 4) Erstellen Sie eine Liste von Sammelstellen (S#) und geben Sie an, wie viele Abgaben pro Sammelstelle gemacht worden sind. Es sollen nur diejenigen Sammelstellungen aufgelistet werden, an die mehr als 5 Abgaben gemacht worden sind und von denen mehr als 3 Fabriklieferungen ausgegangen sind

```
SELECT DISTINCT s.s, COUNT(DISTINCT a.A) AS AbgabenAnzahl
FROM sammelstelle s
JOIN abgabe a ON s.s = a.s
JOIN lieferung l ON l.s = a.s
GROUP BY s.s
HAVING COUNT (DISTINCT a.a) > 5
AND
COUNT (DISTINCT l.f) > 3;
```