

# DATENBANKEN

## WINTERSEMESTER 2024/2025

Prof. Ingo Schmitt

Institut für Informatik, Fakultät 1

Fachgebiet Datenbank- und Informationssysteme

Brandenburgische Technische Universität Cottbus-Senftenberg

### WISSENSKATALOG

ERSTELLER: Ole Matzky

 [Ole.Mkzy](#)   
 [matzkole@b-tu.de](mailto:matzkole@b-tu.de)

BEGLEITMATERIAL:

- ANKI Karteikarten 

(Stand: 9. März 2025 | Es wird kein Anspruch auf Vollständigkeit/Korrektheit erhoben  
Bei Anmerkungen, Verbesserungen etc. bitte an die Ersteller wenden)

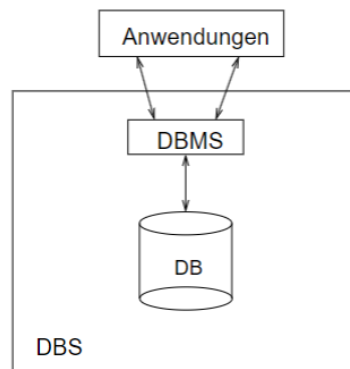
# Inhaltsverzeichnis

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Einführende Begriffe</b>                            | <b>1</b>  |
| <b>2</b> | <b>Entity-Relationship-Modell</b>                      | <b>4</b>  |
| 2.1      | Grundlagen . . . . .                                   | 4         |
| 2.2      | Komplexe Attribute . . . . .                           | 4         |
| 2.3      | Beziehungsmenge . . . . .                              | 5         |
| 2.3.1    | Kardinalität . . . . .                                 | 5         |
| 2.3.2    | Rekursive Beziehungsmenge . . . . .                    | 5         |
| 2.4      | Schwache Entitätenmengen . . . . .                     | 5         |
| 2.5      | Spezialisierung . . . . .                              | 6         |
| 2.6      | Beispielaufgaben . . . . .                             | 6         |
| <b>3</b> | <b>Relationales Datenbankmodell</b>                    | <b>7</b>  |
| 3.1      | Begriffe . . . . .                                     | 7         |
| 3.2      | Transformation ER-Modell in Relationenschema . . . . . | 8         |
| 3.2.1    | Allgemeine Entitätenmenge . . . . .                    | 8         |
| 3.2.2    | Schwache Entitätenmenge . . . . .                      | 8         |
| 3.2.3    | Allgemeine Beziehungsmenge (m:n) . . . . .             | 8         |
| 3.2.4    | Funktionale Abhängigkeiten (1:n, 1:1) . . . . .        | 8         |
| 3.2.5    | Komplexe Attribute . . . . .                           | 8         |
| 3.2.6    | Spezialisierung . . . . .                              | 9         |
| 3.3      | Beispielaufgaben . . . . .                             | 9         |
| <b>4</b> | <b>Datenbankentwurf</b>                                | <b>10</b> |
| 4.1      | Schlüssel . . . . .                                    | 10        |
| 4.2      | Referentielle Integrität . . . . .                     | 10        |
| 4.3      | Funktionale Abhängigkeiten . . . . .                   | 10        |
| 4.3.1    | Superschlüssel . . . . .                               | 11        |
| 4.3.2    | Abschluss . . . . .                                    | 11        |
| 4.3.3    | Armstrong-Axiome . . . . .                             | 11        |
| 4.3.4    | Attribut-Abschluss . . . . .                           | 12        |
| 4.4      | Mehrwertige Abhängigkeiten . . . . .                   | 12        |
| 4.5      | Normalisierung . . . . .                               | 13        |
| 4.5.1    | Verbundtreue . . . . .                                 | 13        |
| 4.5.2    | Abhängigkeitstreue . . . . .                           | 13        |
| 4.5.3    | 1. Normalform (1NF) . . . . .                          | 13        |
| 4.5.4    | 2. Normalform (2NF) . . . . .                          | 14        |
| 4.5.5    | 3. Normalform (3NF) . . . . .                          | 14        |
| 4.5.6    | Boyce-Codd-Normalform (BCNF) . . . . .                 | 15        |
| 4.5.7    | 4. Normalform (4NF) . . . . .                          | 15        |
| 4.6      | Beispielaufgaben . . . . .                             | 15        |
| <b>5</b> | <b>SQL</b>   | <b>16</b> |
| 5.1      | Query Language (QL) . . . . .                          | 16        |
| 5.1.1    | Umbenennung . . . . .                                  | 16        |
| 5.1.2    | Verbund . . . . .                                      | 16        |

|          |   |           |
|----------|---|-----------|
| 5.1.3    | Mengenoperationen . . . . .                     | 17        |
| 5.1.4    | Gruppierung + Aggregatfunktionen . . . . .      | 17        |
| 5.1.5    | Verschachtelte Abfragen . . . . .               | 18        |
| 5.1.6    | Sortierung . . . . .                            | 18        |
| 5.2      | Data Manipulation Language (DML) . . . . .      | 18        |
| 5.2.1    | Einfügen von Daten . . . . .                    | 18        |
| 5.2.2    | Löschen von Daten . . . . .                     | 18        |
| 5.2.3    | Ändern von Daten . . . . .                      | 18        |
| 5.3      | Data Definition Language (DDL) . . . . .        | 18        |
| 5.3.1    | Erstellen von Tabellen . . . . .                | 18        |
| 5.3.2    | Löschen von Tabellen . . . . .                  | 19        |
| 5.3.3    | Ändern von Tabellen . . . . .                   | 19        |
| 5.3.4    | Kopieren von Tabellen . . . . .                 | 19        |
| 5.3.5    | Erstellen von Sichten . . . . .                 | 19        |
| 5.3.6    | Löschen von Sichten . . . . .                   | 19        |
| 5.3.7    | Ändern von Sichten . . . . .                    | 19        |
| 5.4      | Datentypen . . . . .                            | 20        |
| 5.5      | Privilegien . . . . .                           | 20        |
| 5.5.1    | Vergabe . . . . .                               | 20        |
| 5.5.2    | Widerrufen . . . . .                            | 20        |
| 5.6      | Rekursion . . . . .                             | 21        |
| 5.7      | Beispielaufgaben . . . . .                      | 21        |
| <b>6</b> | <b>Datenbanksprachen</b>                        | <b>22</b> |
| 6.1      | Relationale Algebra . . . . .                   | 22        |
| 6.1.1    | Grundoperationen . . . . .                      | 22        |
| 6.1.1.1  | Beispiele . . . . .                             | 23        |
| 6.1.2    | Aggregatfunktionen . . . . .                    | 23        |
| 6.1.3    | Division . . . . .                              | 24        |
| 6.2      | Relationaler Kalkül . . . . .                   | 24        |
| 6.2.1    | Sichere Ausdrücke . . . . .                     | 24        |
| 6.2.2    | Relationale Vollständigkeit . . . . .           | 24        |
| 6.2.3    | Unterschied Tupel- und Bereichskalkül . . . . . | 24        |
| 6.2.4    | Relationaler Tupelkalkül . . . . .              | 25        |
| 6.3      | Beispielaufgaben . . . . .                      | 25        |
| <b>7</b> | <b>Anhang</b>                                   | <b>26</b> |
| 7.1      | Beispiele . . . . .                             | 26        |
| 7.2      | Beweise . . . . .                               | 38        |
| 7.3      | „Wahr oder Falsch“-Fragen . . . . .             | 39        |
| 7.4      | „Wahr oder Falsch“-Lösungen . . . . .           | 53        |

# Kapitel 1 Einführende Begriffe

| Begriff  | Definition  |
|--|---|
| <b>Probleme traditioneller Dateiverwaltung</b> | <ul style="list-style-type: none"><li>• Programm-Daten-Abhängigkeit</li><li>• keine Anfragesprache</li><li>• schlechte Datenmodellierung</li><li>• Redundanz</li></ul>  |
| <b>Programm-Daten-Abhängigkeit</b>             | Daten werden mit Metadaten des Programms abgespeichert → können von anderen Programmen nicht nativ interpretiert werden (Problem!)  |
| <b>DB</b>                                      | Datenbank → Sammlung logisch zusammengehöriger Daten, die persistent (dauerhaft) gespeichert wird (länger als die Lebensdauer eines Anwendungsprozesses)  |
| <b>DBMS</b>                                    | Datenbank-Management-System → Softwarepaket, welches die Datenbank verwaltet (Löschen, Einfügen, Abfragen) und eine Schnittstelle zwischen Anwendung und Datenbank realisiert<br><i>Bsp.:</i> Oracle Database, Microsoft SQL Server, MariaDB, MySQL |
| <b>DBS</b>                                     | Datenbanksystem → kombiniert DBMS mit einer anwendungsspezifischen Datenbank, inklusive der benötigten Metadaten  |



|                        |  |
|------------------------|--|
| <b>Datenmodell</b>     | Sammlung von Konzepten zum Beschreiben von <ul style="list-style-type: none"><li>• Daten und Beziehungen</li><li>• Bedeutung der Daten und Integritätsbedingungen</li><li>• Operationen zur Verhaltensmodellierung</li></ul> |
| <b>Datenbankmodell</b> | spezielles Datenmodell zur Beschreibung einer Datenbank<br><i>Bsp.:</i> hierarchisches DM, Netzwerk - DM, relationales DM, ER-Model, objektorientiertes DM, objekt-relationales DM   |
| <b>Datenbankschema</b> | <ul style="list-style-type: none"><li>• Beschreibung einer Datenbank unter Ausnutzung der Konzepte eines Datenbankmodells</li><li>• definiert Struktur und Einschränkungen einer Datenbank</li></ul>                         |




| Begriff                              | Definition   |
|--------------------------------------|--|
| Zusammenfassend:                     | Datenmodell $\xrightarrow{\text{beschreibt}}$ Datenbankmodell $\xrightarrow{\text{beschreibt}}$ Datenbank-schema   |
| <b>Datenbanksprachen</b>             | <ul style="list-style-type: none"> <li>• <b>Data Definition Language (DDL)</b>: Erstellen und Modifizieren eines Schemas</li> <li>• <b>Data Manipulation Language (DML)</b>: Einfügen, Löschen und Ändern von Daten</li> <li>• <b>Query Language (QL)</b>: Abfragen von Daten</li> <li>• <b>Database Programming Language (DBPL)</b>: Programmieren von Datenbankanwendungen</li> </ul>  |
| <b>Datenbanknutzer</b>               | <ul style="list-style-type: none"> <li>• <b>Endnutzer</b>: verwendet Anwendungsprogramme (Formulare)</li> <li>• <b>DB-Anwendungsprogrammierer</b>: <ul style="list-style-type: none"> <li>– verwendet PL, DBPL, DML, QL</li> <li>– entwirft Formulare</li> <li>– erzeugt und verwendet Nutzersichten</li> <li>– implementiert Anwendungen</li> </ul> </li> <li>• <b>Datenbankadministrator</b>: <ul style="list-style-type: none"> <li>– verwendet DDL, DML, QL</li> <li>– entwirft logische/physische Schemata</li> <li>– entwirft Integritätsbedingungen</li> <li>– kümmert sich um Sicherheit, Autorisierung und Verfügbarkeit der Daten (Sicherung und Wiederherstellung)</li> <li>– überwacht Leistung</li> <li>– führt Tuning durch</li> </ul> </li> </ul> |
| <b>3-Ebenen-Schema-Architektur</b>   | Beschreibung der Architektur eines Datenbank-Management-Systems in drei getrennte Ebenen: <ul style="list-style-type: none"> <li>• Externe Schemata</li> <li>• Konzeptuelles Schema</li> <li>• Internes Schema</li> </ul>  |
| <i>Externes Schema</i>               | verschiedene Nutzersichten (definiert auf konzeptuellem Schema) für verschiedene Anwendungen, meist Datenbankausschnitt  |
| <i>Konzeptuelles Schema</i>          | Beschreibung der gesamten Datenbank (Struktur, Integritätsbedingungen, Autorisierung, ...) unter Nutzung eines Datenbankmodells, unabhängig von Implementierungsdetails  |
| <i>Internes Schema</i>               | physische Darstellung der Datenbank, abhängig vom Betriebssystem (z.B. Indices anlegen)  |
| <b>Physische Datenunabhängigkeit</b> | Unabhängigkeit von Implementierungsdetails → konzeptuelles Schema ist unabhängig von Änderungen/Implementierung des internen Schemas   |

| Begriff                             | Definition   |
|-------------------------------------|--|
| <b>Logische Datenunabhängigkeit</b> | <p>Entkopplung von externen und konzeptuellen Schema:</p> <ul style="list-style-type: none"><li>• externe Schemata unabhängig von einigen Modifikationen des konzeptuellen Schemas</li><li>• Modifikationen externer Schemata möglich ohne konzeptuelles Schema modifizieren zu müssen</li></ul> <pre>graph TD; ES1[externes Schema 1] --&gt; KS[konzeptuelles Schema]; ES2[externes Schema 2] --&gt; KS; KS --&gt; IS[internes Schema];</pre> |
| <b>Codd'schen Regeln</b>            | <ol style="list-style-type: none"><li>1. Integration</li><li>2. Operationen</li><li>3. Katalog</li><li>4. Nutzersichten</li><li>5. Konsistenzüberwachung</li><li>6. Zugriffskontrolle</li><li>7. Transaktionen</li><li>8. Synchronisation</li><li>9. Datensicherung und Wiederherstellung</li></ol>  |
| <b>Integration</b>                  | Einheitliche Datenverwaltung und keine Redundanz   |
| <b>Katalog</b><br>(Data Dictionary) | enthält Metadaten (Daten über Daten) → Schemadaten über die Datenbankstruktur  |
| <b>Nutzersichten</b>                | verschiedene Sichten auf die Datenbank (virtuelle Relationen)  |
| <b>Konsistenz</b>                   | Korrektheit der Daten, in Bezug dass alle Kopien im System gleich sind   |
| <b>Integrität</b>                   | Semantische Korrektheit und Vollständigkeit der Daten  |
| <b>Zugriffskontrolle</b>            | Zugriff auf Datenbank nur für berechnigte/autorisierte Nutzer  |
| <b>Transaktion</b>                  | atomare, logische Einheit von Datenbankoperationen, die die Datenbank von einem konsistenten Zustand in einen konsistenten, eventuell veränderten, Zustand überführt (wird komplett oder gar nicht ausgeführt)   |
| <b>Synchronisation</b>              | Vermeidung von Inkonsistenzen aufgrund von Multi-User-Zugriff im Mehrbenutzerbetrieb (konkurrierender Zugriff)   |
| <b>Datensicherung</b>               | regelmäßige Sicherung der Datenbank, um Datenverlust zu vermeiden (bspw. bei Stromausfall)   |


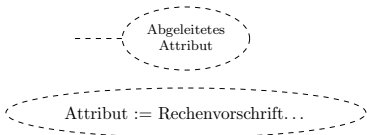
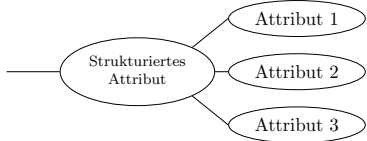

# Kapitel 2 Entity-Relationship-Modell

Entwurfsdatenmodell zur konzeptuellen Modellierung eines Datenbank-Entwurfs, aber kein Datenbankmodell!

## 2.1 Grundlagen

|                        | Beschreibung  | Notation  |
|------------------------|---|---|
| <b>Entitätenmenge</b>  | Menge von Entitäten (wahrnehmbares Objekt) gleichen Typs, die sich gleiche Eigenschaften teilen                                   |  |
| <b>Beziehungsmenge</b> | Sammlung gleicher Beziehungen zwischen Entitäten  |  |
| <b>Attribute</b>       | Eigenschaft einer Menge von Entitäten oder Beziehungen mit zugehörigem Wertebereich<br><br>Primärschlüssel → Namen unterstrichen! |  |

## 2.2 Komplexe Attribute

|                                | Beschreibung  | Notation  |
|--------------------------------|---|---|
| <b>Mehrwertiges Attribut</b>   | enthält Menge von Werten<br><br>Bsp.: Email-Adresse<br>(kann man mehrere haben)   |  |
| <b>Abgeleitetes Attribut</b>   | Wert wird aus anderen, nicht-abgeleiteten Attributen anhand einer Rechenvorschrift bestimmt<br><br>Bsp.: Alter (aus Geburtsdatum berechnet) |  |
| <b>Strukturiertes Attribut</b> | Wert entspricht der Verkettung der Unterattribute<br><br>Bsp.: Adresse (Straße, PLZ, Ort)   |  |
| <b>Optionales Attribut</b>     | Wert nicht für jede Entität vorhanden (NULL-Werte)<br><br>Bsp.: Zweitname (nicht jeder hat einen)   |  |

## 2.3 Beziehungsmenge

**Arität, Grad** → Anzahl der beteiligten Entitäten

- **Binäre Beziehungsmenge** → 2 Entitäten
- **Ternäre Beziehungsmenge** → 3 Entitäten
- **N-äre Beziehungsmenge** → n Entitäten

Zerlegung ternärer → 3× binär → Informationsverlust ↯

Beziehungsmengen → 3× binär + künstliche Entität in der Mitte → ✓

### Beispiel Zerlegung ternärer Beziehungsmenge

#### 2.3.1 Kardinalität

Kardinalität beschränkt die Anzahl der beteiligten Entitäten einer Beziehungsmenge.



→  $\min, \max \in \{0, 1, \dots, *\}$ ,  $\min \leq \max$  (\* = beliebig viele)

### Beispiel zu 2.3.1 Kardinalität

#### 2.3.2 Rekursive Beziehungsmenge

Beziehungsmenge, die eine Entitätenmenge mehrfach involviert

→ Rollennamen zur Unterscheidung

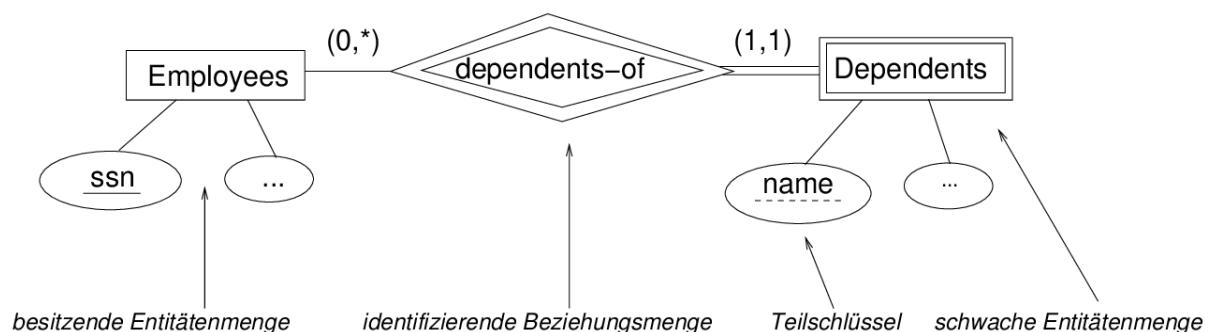
### Beispiel zu 2.3.2 Rekursive Beziehungsmenge

## 2.4 Schwache Entitätenmengen

Entitätenmenge, die von einer anderen Entitätenmenge abhängig ist (alles doppelt umrahmt)

→ Primärschlüssel besteht aus Primärschlüssel der starken Entität und eigenem Teilschlüssel (gestrichelt unterstrichen)

→ Kardinalität 1:1 oder 1:n



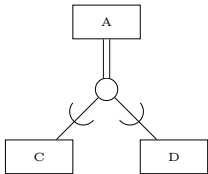
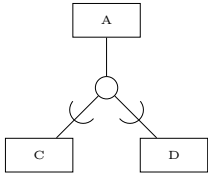
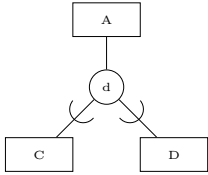
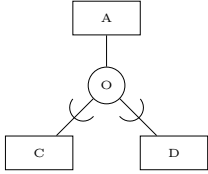


## 2.5 Spezialisierung

Entitätenmengen können spezialisiert werden

→ Unterteilung in Super- und Sub-Entitätenmengen

→ vermeidet redundante Attribute

|                    | Beschreibung   | Notation  |
|--------------------|--|---|
| <b>total</b>       | Es gibt keine mögliche Entität, die keiner Spezialisierung angehört<br>(Doppellinie bei Super-Entitätenmenge)        |    |
| <b>partiell</b>    | Es können Entitäten existieren, die in keiner Spezialisierung vorkommen<br>(einfache Linie bei Super-Entitätenmenge) |    |
| <b>disjunkt</b>    | Entitäten können nur einer Spezialisierung angehören<br>(„d“ im Kreis)   |   |
| <b>überlappend</b> | Entitäten können mehreren Spezialisierungen angehören<br>(„o“ im Kreis)  |  |

### Beispiel zu 2.5 Spezialisierung

## 2.6 Beispielaufgaben

W | F 1

W | F 2

W | F 3

W | F 4

W | F 5

W | F 6

W | F 7

# Kapitel 3    *Relationales Datenbankmodell*

## 3.1    Begriffe

| Begriff                             | Beschreibung  |
|-------------------------------------|---|
| <b>Relation (informell)</b>         | Beziehung zwischen Objekten   |
| <b>Relation (Mathematik)</b>        | $R \subseteq A \times B$ (Teilmenge des kartesischen Produkts)<br>→ zugrundeliegende Eigenschaft bestimmt, welche<br>Tupel Teil der Relation sind   |
| <b>Relationenschema</b>             | beschreibt den Aufbau einer Relation ( $\rightsquigarrow$ Tabellenkopf)<br>$R(A_1, \dots, A_n)$ <ul style="list-style-type: none"><li>• <math>R \rightarrow</math> Name der Relation</li><li>• <math>A_1, \dots, A_n \rightarrow</math> Attributliste</li><li>• Funktion 'dom' ordnet Attribut <math>A_i</math> Wertebereich <math>D_i</math> zu (<math>D_i = \text{dom}(A_i)</math>)</li><li>• Grad <math>\rightarrow</math> Anzahl der Attribute (<math>n</math>)</li></ul> |
| <b>Relation</b>                     | konkrete Tabelle eines Relationenschemas<br>$R(A_1, \dots, A_n)$<br>$r(R)$ <ul style="list-style-type: none"><li>• <math>r = \{t_1, \dots, t_m\} \rightarrow</math> Menge von <math>n</math>-Tupel</li><li>• Kardinalität <math>\rightarrow</math> Anzahl der Tupel (<math>m</math>)</li></ul>  |
| <b>Tupel</b>                        | geordnete Liste von Attributwerten ( $\rightsquigarrow$ Zeile)<br>$n$ -Tupel $t = \{v_1, \dots, v_n\}$ <ul style="list-style-type: none"><li>• <math>t[A_i]</math> oder <math>t.A_i \rightarrow i</math>-ter Wert des Tupels</li><li>• <math>t[X] \rightarrow</math> Wertekombination bzgl. Attributmenge <math>X = \{A_i, A_j, \dots\}</math></li></ul>  |
| <b>Relationales Datenbankschema</b> | Menge von Relationenschemata<br>$S = \{R_1, \dots, R_m, IB\}$ <ul style="list-style-type: none"><li>• <math>R_i \rightarrow</math> Relationenschema</li><li>• <math>IB \rightarrow</math> Integritätsbedingungen</li></ul>  |
| <b>Relationale Datenbank</b>        | Menge von Relationen, konkreter Zustand eines relationalen Datenbankschemas $S = \{R_1, \dots, R_m, IB\}$<br>$DB = \{r_1, \dots, r_m\}$ mit $r_i(R_i)$ und Integritätsbedingungen erfüllt<br><br>→ Datenbankschema gewöhnlich fest, Datenbank zeitlich variabel   |

## 3.2 Transformation ER-Modell in Relationenschema

### 3.2.1 Allgemeine Entitätenmenge

starke (nicht schwache) Entitätenmenge  $E$  mit Attributen  $A_1, \dots, A_n$  und Schlüsselattributen  $X_1, \dots, X_k$ :

$$E(\underline{X_1, \dots, X_k}, A_1, \dots, A_n)$$

Beispiel zu 3.2.1 Allgemeine Entitätenmenge

### 3.2.2 Schwache Entitätenmenge

$E_1$  ist schwache Entitätenmenge mit Schlüssel  $X_1$ , und  $E_2$  ist besitzende (starke) Entitätenmenge mit Schlüssel  $X_2$ :

- $E_1(\underline{X_2} \rightarrow E_2, X_1, \dots)$  (Fremdschlüssel  $X_2$  von  $E_2$ )
- $E_2(\underline{X_2}, \dots)$

Beispiel zu 3.2.2 Schwache Entitätenmenge

### 3.2.3 Allgemeine Beziehungsmenge (m:n)

Beziehungsmenge  $R$  mit Attributen  $A_1, \dots, A_m$  involviert Entitätenmengen  $E_1, \dots, E_n$  und  $X_i$  sei Schlüssel von Entitätenmenge  $E_i$ :

$$R(\underline{X_1 \rightarrow E_1, \dots, X_n \rightarrow E_n}, A_1, \dots, A_m)$$

Beispiel zu 3.2.3 Allgemeine Beziehungsmenge (m:n)

### 3.2.4 Funktionale Abhängigkeiten (1:n, 1:1)

Beziehungsmenge  $R$  mit Attributen  $A_1, \dots, A_m$  involviert Entitätenmengen  $E_1$  und  $E_2$  (mit entsprechenden Schlüsseln  $X_1$  und  $X_2$ ), wobei jede Entität von  $E_1$  in  $R$  höchstens mit einer Entität von  $E_2$  assoziiert ist (n:1):

- $E_2(\underline{X_2}, \dots, A_1, \dots, A_m, X_1 \rightarrow E_1)$  (Erweitern um Attribute von  $R$  und Fremdschlüssel  $X_1$  von  $E_1$ )
- **kein** Relationenschema für  $R$ !

Beispiel zu 3.2.4 Funktionale Abhängigkeiten (1:n, 1:1)

### 3.2.5 Komplexe Attribute

---

|                              | Transformation   |
|------------------------------|--|
| <b>Mehrwertiges Attribut</b> | Entitätenmenge $E$ mit Attributen $A_1, \dots, A_n$ , Schlüssel $X$ und mehrwertigen Attribut $B$ : <ul style="list-style-type: none"> <li>• <math>E(\underline{X}, A_1, \dots, A_m)</math> (<math>E</math> ohne <math>B</math>)</li> <li>• <math>E - B(\underline{B}, X \rightarrow E)</math> (neue Relation für <math>B</math>)</li> </ul> |

---

|                                |   |
|--------------------------------|---|
| <b>Strukturiertes Attribut</b> | Entitätenmenge $E$ mit Schlüssel $X$ und strukturiertem Attribut $A$ , mit wiederum Unterattributen $A_1, \dots, A_n$ :<br>$E(\underline{X}, A_1, \dots, A_n)$ (ohne $A$ )  |
| <b>Abgeleitetes Attribut</b>   | Entitätenmenge $E$ mit Attributen $A_1, \dots, A_n$ , Schlüssel $X$ und abgeleitetem Attribut $B$ : <ul style="list-style-type: none"> <li>• <math>E(\underline{X}, A_1, \dots, A_m, B)</math> (mit <math>B</math> als Integritätsbedingung)</li> <li>• Definieren einer Nutzersicht</li> </ul> |
| <b>Optionales Attribut</b>     | Entitätenmenge $E$ mit Attributen $A_1, \dots, A_n$ , Schlüssel $X$ und optionalem Attribut $B$ :<br>$E(\underline{X}, A_1, \dots, A_m, B)$ ( $B$ einfach übernehmen, NULL-Werte zulässig)  |

### Beispiel Transformation Mehrwertiges Attribut

#### 3.2.6 Spezialisierung

Für Superklasse  $C$  mit Schlüssel  $X$  und Attributen  $A_1, \dots, A_n$  sowie  $m$  Subklassen  $S_1, \dots, S_m$ :

- $Super(\underline{X}, A_1, \dots, A_n)$  ( $C$ , ohne spezifische Attribute der Subklassen)
- $Sub_i(\underline{X} \rightarrow Super, \dots)$  (mit spezifischen Attributen der Subklasse  $S_i$ )

### Beispiel zu 3.2.6 Spezialisierung

#### weitere Optionen der Transformation

### 3.3 Beispielaufgaben

W | F 8

W | F 9

W | F 10

W | F 11

W | F 12

W | F 13

W | F 14

# Kapitel 4 Datenbankentwurf

## 4.1 Schlüssel

|                          |   |
|--------------------------|---|
| <b>Superschlüssel</b>    | Menge von Attributen, deren Werte jede Entität eindeutig bestimmt<br>$SK \subseteq \{A_1, \dots, A_n\}$ ist Superschlüssel für die Relation $r(R) = \{A_1, \dots, A_n\}$ , wenn gilt:<br>$\forall t_1, t_2 \in r : t_1 \neq t_2 \implies \exists A_i \in SK : t_1[A_i] \neq t_2[A_i]$ |
| <b>Schlüsselkandidat</b> | Minimaler Superschlüssel $\rightarrow$ kein Attribut kann entfernt werden, ohne dass die Eindeutigkeitsbedingung verletzt wird  |
| <b>Primärschlüssel</b>   | Ausgewählter Schlüsselkandidat. NULL-Werte dürfen nicht auftreten.  |
| <b>Fremdschlüssel</b>    | Verweis auf Primärschlüssel einer anderen Entitätenmenge  |

### Beispiel Primärschlüssel

## 4.2 Referentielle Integrität

Jeder Fremdschlüsselwert verweist auf existierende Tupel der referenzierten Relation oder ist NULL:

$$\forall t_1 \in r_1(R_1) \in DB : t_1[FK] = \text{NULL} \vee \exists t_2 \in r_2(R_2) : t_2[PK] = t_1[FK]$$

und sie dürfen nicht durch Löschen oder Updates korrumpiert werden.

## 4.3 Funktionale Abhängigkeiten

### Definition

Für ein Relationenschema  $R(A_1, \dots, A_n)$  und  $X, Y \subseteq \{A_1, \dots, A_n\}$  gilt:  
Es besteht die funktionale Abhängigkeit **FD**:  $X \rightarrow Y$ , wenn für alle Tupel  $t_1, t_2 \in r(R)$  (in einem korrekten Datenbankzustand) gilt:

$$t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y]$$

(wenn 2 Tupel in  $X$  übereinstimmen, dann auch in  $Y$ )

Sprechweise:  $X$  bestimmt Funktional  $Y$ , oder  $Y$  ist funktional abhängig von  $X$

### Bemerkung

Für  $Y \subseteq X$  gilt immer die funktionale Abhängigkeit **FD**:  $X \rightarrow Y$  (**triviale Abhängigkeit**)

### 4.3.1 Superschlüssel

- Wenn  $X$  Superschlüssel von  $R(A_1, \dots, A_n)$  ist, dann gilt  $X \rightarrow Y$  für alle  $Y \subseteq \{A_1, \dots, A_n\}$
- Wenn  $X \rightarrow A_i$  für jedes Attribut  $A_i$  von  $R(A_1, \dots, A_n)$  gilt, dann ist  $X$  ein Superschlüssel

### 4.3.2 Abschluss

#### Definition

- FD  $X \rightarrow Y$  ist abgeleitet aus  $F$  (Menge funktionaler Abhängigkeiten), wenn  $X \rightarrow Y$  in jedem Datenbankzustand erfüllt ist, der auch  $F$  erfüllt

Notation:  $F \models X \rightarrow Y$

- Der **Abschluss**  $F^+$  einer Menge funktionaler Abhängigkeiten  $F$  ist die Menge aller funktionalen Abhängigkeiten, die aus  $F$  abgeleitet werden können.

$$F^+ := \{f \mid f \text{ ist FD} \wedge F \models f\}$$

- zwei Mengen funktionaler Abhängigkeiten  $E$  und  $F$  sind **äquivalent**, wenn  $E^+ = F^+$  gilt

#### Beispiel zu 4.3.2 Abschluss

### 4.3.3 Armstrong-Axiome

#### Armstrong-Axiome

1. **Reflexivität:**  $Y \subseteq X \implies X \rightarrow Y$  (triviale Abhängigkeit)
  2. **Erweiterung:**  $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
  3. **Transitivität:**  $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
- Daraus ableitbar:
4. **Vereinigung:**  $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$
  5. **Zerlegung:**  $\{X \rightarrow YZ\} \models X \rightarrow Y, X \rightarrow Z$
  6. **Pseudotransitivität:**  $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$

#### Beweis der Gültigkeit der abgeleiteten Regeln

#### Bemerkung

- $XYZ \rightarrow UV$  ist Kurzschreibweise von  $\{X, Y, Z\} \rightarrow \{U, V\}$  (Mengen  $\rightarrow$  keine Reihenfolge), daher spielt die Reihenfolge der Attribute in den Regeln keine Rolle!
- Aus den Armstrong-Axiomen folgt:
  - Man kann immer **rechts wegnehmen** (Zerlegung)
  - Man kann immer **links hinzufügen** (Erweiterung + Zerlegung)

### 4.3.4 Attribut-Abschluss

#### Definition

Der **Attribut-Abschluss**  $X^+$  eines Attributsets  $X$  bzgl. einer Menge funktionaler Abhängigkeiten  $F$  ist die Menge aller Attribute, die funktional von  $X$  abhängen:

$$X^+ := \{A \mid X \rightarrow A \in F^+\}$$

Es gilt:

$$X \rightarrow U \in F^+ \Leftrightarrow \underbrace{U \subseteq X^+}_{\text{effizient}} \text{ (bzgl. } F)$$

Es folgt:

$$F^+ = G^+ \text{ wenn gilt: } \begin{aligned} &\forall A \rightarrow B \in F^+ : \{B\} \subseteq \{A\}^+ \text{ bzgl. } G \\ &\wedge \forall A \rightarrow B \in G^+ : \{B\} \subseteq \{A\}^+ \text{ bzgl. } F \end{aligned}$$

#### Algorithmus zur Berechnung des Attribut-Abschlusses

1. Initialisiere  $X^+ := X$  (Reflexivität:  $X \rightarrow X \implies X \subseteq X^+$ )
2. Solange sich  $X^+$  ändert:
  - $\hookrightarrow$  Für jede FD  $Y \rightarrow Z$  in  $F$ :
  - $\hookrightarrow$  Wenn  $Y \subseteq X^+$ , dann füge  $Z$  zu  $X^+$  hinzu (Transitivität)

#### Beispiel zu 4.3.4 Attribut-Abschluss

## 4.4 Mehrwertige Abhängigkeiten

#### Definition

- Eine mehrwertige Abhängigkeit  $X \twoheadrightarrow Y$  ist eine funktionale Zuordnung von Wertemengen zu Werten. Die von der Zuordnung nicht betroffenen (unabhängigen) Attribute  $R \setminus (X \cup Y)$  sind dann aber immer gleich und erzeugen Redundanz.

Formal: MVD  $X \twoheadrightarrow Y$  in Relation  $R$  ist erfüllt, wenn

$$\begin{aligned} \forall t_1, t_2 \in r : (t_1 \neq t_2 \wedge t_1[X] = t_2[X]) &\implies \exists t_3 \in r : t_3[X] = t_1[X] \\ &\wedge \underbrace{t_3[Y] = t_1[Y] \wedge t_3[R \setminus (X \cup Y)] = t_2[R \setminus (X \cup Y)]}_{\text{überkreuz}} \end{aligned}$$

- Jede funktionale Abhängigkeit ist auch eine mehrwertige Abhängigkeit (aber nicht umgekehrt).
- Eine MVD  $X \twoheadrightarrow Y$  ist *trivial*, wenn  $Y \subseteq X$  oder  $X \cup Y = R$  ist.

#### Beispiel zu 4.4 Mehrwertige Abhängigkeiten

## 4.5 Normalisierung

### Ziel

- minimale Redundanzen
- minimale Anomalien bei Einfügen, Löschen, Ändern

→ **Normalisierung** durch **Zerlegung**

→ Zerlegung muss **Verbundtreue** und **Abhängigkeitstreue** gewährleisten

### 4.5.1 Verbundtreue

#### Definition

Verbund (inverse Operation zur Zerlegung) zerlegter Relationen muss immer die ursprüngliche Relation ergeben!

**Testkriterium:** Die Zerlegung des Relationschema  $R$  in die Relationschemata  $R_1$  und  $R_2$  ist verlustfrei (und somit **verbundtreu**), wenn gilt ( $F$  Menge der FDs von  $R$ ):

$$R_1 \cap R_2 \rightarrow R_1 \in F^+ \vee R_1 \cap R_2 \rightarrow R_2 \in F^+$$

#### Beispiel zu 4.5.1 Verbundtreue

### 4.5.2 Abhängigkeitstreue

#### Definition

Die Zerlegung des Relationschema  $R$  in die Relationschemata  $R_1, R_2, \dots, R_p$  ist **abhängigkeitstreue**, wenn gilt ( $F$  Menge der FDs von  $R$ ):

$$(\pi_{R_1}(F) \cup \dots \cup \pi_{R_p}(F))^+ = F^+$$

mit  $\pi_{R_i}(F) = \{A \rightarrow B \in F^+ \mid A \cup B \subseteq R_i\}$  (alle FD, die man aus  $F$  ableiten kann, und wo die Attributmengen  $A$  und  $B$  in  $R_i$  enthalten sind)

**Testkriterium:**

$$F^+ = G^+ \text{ wenn gilt: } \forall A \rightarrow B \in F^+ : \{B\} \subseteq \{A\}^+ \text{ bzgl. } G \\ \wedge \forall A \rightarrow B \in G^+ : \{B\} \subseteq \{A\}^+ \text{ bzgl. } F$$

#### Beispiel zu 4.5.2 Abhängigkeitstreue

### 4.5.3 1. Normalform (1NF)

#### Definition

Eine Relation  $r(R)$  ist in **1. Normalform**, wenn jedes Attribut atomar ist (keine mehrwertigen Attribute) und alle Werte atomar sind (keine mehrwertigen Tupel).

#### Umsetzung

- Zerlegung von mehrwertigen Attributen in eigene Relationen (ähnlich Transformation mehrwertiger Attribute vom ER-Modell ins Relationenschema)



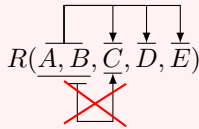
- Zerlegung von mehrwertigen Tupeln in eigene Tupel (pro Element eine Zeile)

### Beispiel zu 4.5.3 1. Normalform (1NF)

#### 4.5.4 2. Normalform (2NF)

##### Definition

Eine Relation ist in **2. Normalform**, wenn sie in 1NF ist, und es **keine** funktionale Abhängigkeit von einem Teil des Schlüssels auf ein Nicht-Schlüsselattribut gibt.



##### Umsetzung

- Zerlegung in 2 Relationen, so dass jede Relation nur vollständige Abhängigkeiten enthält
- Teilschlüssel wird Primärschlüssel der ausgelagerten Relation

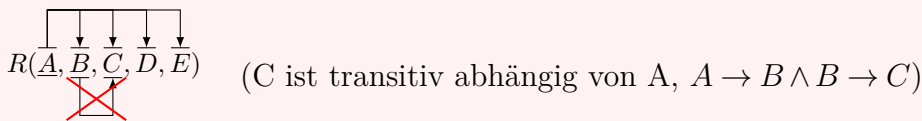
### Beispiel zu 4.5.4 2. Normalform (2NF)

#### 4.5.5 3. Normalform (3NF)

##### Definition

Eine Relation  $R$  ist in 3NF, wenn sie in 2NF ist, und **kein** Non-Primattribut *transitiv abhängig* von einer Teilmenge eines Schlüsselst ist. (keine funktionale Abhängigkeiten von Non-Prim zu Non-Prim)

**transitive Abhängigkeit:**  $FD X \rightarrow Y$  ist eine transitive Abhängigkeit, wenn eine Attributmenge  $Z$  existiert (keine Teilmenge von Schlüsselkandidat oder Schlüssel von  $R$ ) und  $X \rightarrow Z$  sowie  $Z \rightarrow Y$  (nichttrivial) gelten.



##### Umsetzung

- Jede Non-Prim  $\rightarrow$  Non-Prim FD in eigene Relation
- Non-Primattribut wird Primärschlüssel der ausgelagerten Relation

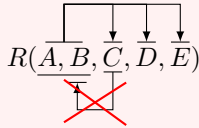
### Beispiel zu 4.5.5 3. Normalform (3NF)

### 4.5.6 Boyce-Codd-Normalform (BCNF)

#### Definition

Eine Relation  $R$  ist in BCNF, wenn sie in 2NF ist, und für **jede** funktionale Abhängigkeit  $X \rightarrow Y$  in  $R$  gilt:

- $X \rightarrow Y$  ist trivial ( $Y \subseteq X$ )  
oder
- $X$  ist Superschlüssel von  $R$



#### Umsetzung

- Zerlegung in 2 Relationen, so dass jede Relation nur vollständige Abhängigkeiten enthält
- ehemaliges Non-Prim-Attribut wird Primärschlüssel der ausgelagerten Relation

### Beispiel zu 4.5.6 Boyce-Codd-Normalform (BCNF)

#### Bemerkung

1NF, 2NF und 3NF sind immer abhängigkeitsreu realisierbar, BCNF **nicht immer!** (Abhängigkeitsreue wird manchmal unweigerlich verletzt)

Beweis, dass BCNF nicht immer abhängigkeitsreu realisierbar ist

### 4.5.7 4. Normalform (4NF)

#### Definition

Eine Relation  $R$  ist in 4NF, wenn sie in BCNF ist und **jede** mehrwertige Abhängigkeit  $X \twoheadrightarrow Y$  in  $R$

- trivial ist ( $Y \subseteq X$  oder  $X \cup Y = R$ )
- $X$  ein Superschlüssel von  $R$  ist.

#### Umsetzung

$R(\underline{X}, Y, Z)$  mit MVD  $X \twoheadrightarrow Y$  kann verlustfrei in  $R_1(\underline{X}, Y)$  und  $R_2(\underline{X}, Z)$  zerlegt werden

### Beispiel zu 4.5.7 4. Normalform (4NF)

## 4.6 Beispielaufgaben

W | F 15

W | F 16

W | F 17

W | F 18

W | F 19

W | F 20

W | F 21

W | F 22

W | F 23

# Kapitel 5 SQL

## Bemerkung

SQL (Structured Query Language) ist **nicht** Case-Sensitive und benötigt keine Zeilennumbrüche, diese sind aber für bessere Lesbarkeit empfehlenswert.  
(Case-Sensitive → Groß- o. Kleinschreibung relevant)  
Jedes SQL-Statement endet mit einem Semikolon (;).

## Beispieldatenbank aus der Vorlesung

### 5.1 Query Language (QL)

Allgemeine Syntax:

```
SQL
SELECT <Tabelle>.<Spalte1>, <Tabelle>.<Spalte2>, ...
FROM <Tabelle>
WHERE <Bedingung>;
```

- SELECT \* → alle Spalten, SELECT DISTINCT → ohne Duplikate
- In der SELECT-Klausel können auch feste Werte (z.B. SELECT 'Toast'), Berechnungen (z.B. SELECT (<Spalte1> + 3\*<Spalte2>)/12), oder Unterabfragen (Ergebnis darf nur eine Zeile enthalten!) stehen
- WHERE → filter Zeilen (nur Zeilen, wo Bedingung TRUE wird, werden zurückgegeben), kein WHERE → alle Zeilen
- <Bedingung> enthält:
  - Vergleichsoperatoren: =, <, >, <=, >=, <> (ungleich)
  - Logische Operatoren: AND, OR, NOT, IS NULL, IS NOT NULL
  - <Spalte> BETWEEN <MIN> AND <MAX> (inklusive der Grenzen), ROWNUM (Zeilenummer)

#### 5.1.1 Umbenennung

```
SQL
SELECT T.<Spalte1> AS Neuer_Name, T.<Spalte2> AS Neuer_Name
FROM <Tabelle> AS T;
```

- Wenn 2 Tabellen den gleichen Spaltennamen haben, oder eine Tabelle mehrmals vorkommt (Entfernen von Duplikaten mit T1.PK<T2.PK)
- Oracle → ohne AS bei den Tabellen

#### 5.1.2 Verbund

```
SQL
SELECT <Tabelle1>.<Spalte1>, <Tabelle2>.<Spalte2>
FROM <Tabelle1>, <Tabelle2>
WHERE <Tabelle1>.<Spalte1> = <Tabelle2>.<Spalte2>;
```

→ Kartesisches Produkt (alle Kombinationen), WHERE-Bedingung  $\hat{=}$  Verbund-Bedingung

SQL

```
SELECT <Tabelle1>.<Spalte1>, <Tabelle2>.<Spalte2>
FROM <Tabelle1> <Verbund> <Tabelle2> ON <Verbund_Bedingung>
WHERE <Bedingung>;
```

- INNER JOIN, NATURAL JOIN (kein ON <Bedingung>, sondern gleiche Spaltennamen)
- Auffüllen mit NULL-Werten für Zeilen ohne Match: LEFT OUTER JOIN (alle Zeilen aus 1. Tabelle), RIGHT OUTER JOIN (alle Zeilen aus 2. Tabelle), FULL OUTER JOIN (alle Zeilen)
- Oracle → ohne INNER und OUTER

## Bildliche Darstellung

### 5.1.3 Mengenoperationen

SQL

```
(Abfrage 1)
<Mengenoperation>
(Abfrage 2);
```

- UNION (Vereinigung), INTERSECT (Schnittmenge), EXCEPT (Differenzmenge) (MINUS in Oracle)
- Entfernen Duplikate! (außer UNION ALL)
- Müssen vereinigungskompatibel sein (gleiche # Spalten, gleiche Datentypen)

### 5.1.4 Gruppierung + Aggregatfunktionen

SQL

```
SELECT <Spalte1>,<Spalte2>,<Aggregatfunktion>(<Spalte2>)
FROM <Tabelle>
WHERE <Bedingung auf Zeile>
GROUP BY (<Spalte1>,<Spalte2>)
HAVING <Bedingung auf Gruppe>;
```

- Aggregatfunktion → berechnet für jede Gruppe einen Wert (bzw. eine Zeile), z.B.:
  - COUNT(\*) → # Anzahl Zeilen der Tabelle, COUNT(<Spalte>) → # Zeilen ohne NULL in <Spalte>, COUNT(DISTINCT <Spalte>) → ohne Duplikate
  - SUM, AVG, MIN, MAX
- GROUP BY (<Spalte1>,<Spalte2>) → Gruppierung aller Zeilen mit gleichen Werten in <Spalte1> und <Spalte2>
 

(kein GROUP BY → alle Zeilen in einer Gruppe)
- HAVING <Bedingung auf Gruppe> → Filtert Gruppen, Aggregatfunktionen können hier verwendet werden
- Selektion muss alle Gruppierungsspalten und Aggregatfunktionen enthalten, und nichts weiter!
- Gruppierung findet nach der Filterung von WHERE statt!

### 5.1.5 Verschachtelte Abfragen

|   |  |  |
|---|--|--|
| SELECT <Spalte1><br>FROM <Tabelle><br>WHERE <Spalte2> IN<br>(Unterabfrage); | SELECT <Spalte1><br>FROM <Tabelle><br>WHERE <Spalte2> <op> ALL<br>(Unterabfrage);                    | SELECT <Spalte1><br>FROM <Tabelle><br>WHERE EXISTS<br>(Unterabfrage);                              |
| Wert muss in der Unterabfrage vorkommen                                     | Vergleich mit allen Werten der Unterabfrage,<br>ALL → alle müssen TRUE ergeben,<br>ANY → mind. einer | EXISTS → prüft ob Unterabfrage mindestens eine Zeile enthält,<br>UNIQUE → prüft ob keine Duplikate |

### 5.1.6 Sortierung

|  |
|--|
| SQL  |
| SELECT <Spalte1>,<Spalte2><br>FROM <Tabelle><br>WHERE <Bedingung><br>ORDER BY <Spalte1> ASC, <Spalte2> DESC; |

ASC → aufsteigend (kleinster Wert oben), DESC → absteigend (größter Wert oben)

## 5.2 Data Manipulation Language (DML)

### 5.2.1 Einfügen von Daten

|  |
|--|
| SQL  |
| INSERT INTO <Tabelle> (<Spalte1>, <Spalte3>, <Spalte5>)<br>VALUES (<Wert1>, <Wert3>, <Wert5>); |

### 5.2.2 Löschen von Daten

|  |
|--|
| SQL                                      |
| DELETE FROM <Tabelle> WHERE <Bedingung>; |

keine Bedingung → löscht alle Zeilen

### 5.2.3 Ändern von Daten

|  |
|--|
| SQL  |
| UPDATE <Tabelle><br>SET <Spalte1> = <Wert1>, <Spalte2> = <Wert2><br>WHERE <Bedingung>; |

## 5.3 Data Definition Language (DDL)

### 5.3.1 Erstellen von Tabellen

|  |
|--|
| SQL  |
| CREATE TABLE <Tabelle> (<br><Spaltenname 1> <Datentyp> DEFAULT <Wert>,<br><Spaltenname 2> <Datentyp>,<br><Spaltenname 3> <Datentyp> NOT NULL,<br><Spaltenname 4> <Datentyp> CHECK (<Bedingung>), |

```

UNIQUE (<Spaltenname 3>),
PRIMARY KEY (<Spaltenname 1>),
FOREIGN KEY (<Spaltenname 2>) REFERENCES <AndereTabelle>(<Attribut>)
ON DELETE CASCADE
);

```

- Default → Standardwert
- Not Null → Wert darf nicht leer sein
- Check → Bedingung
- Unique → Wert darf nur einmal vorkommen
- On Delete Cascade → Löschen der Zeile, wenn die referenzierte Zeile gelöscht wird
- On Delete Set Null → Setzt den Wert auf NULL, wenn die referenzierte Zeile gelöscht wird

### 5.3.2 Löschen von Tabellen

```

DROP TABLE <Tabelle>;

```

### 5.3.3 Ändern von Tabellen

```

ALTER TABLE <Tabelle> ADD <Spalte> <Datentyp>;
ALTER TABLE <Tabelle> DROP COLUMN <Spalte>;
ALTER TABLE <Tabelle> ALTER COLUMN <Spalte> <Datentyp>;

```

### 5.3.4 Kopieren von Tabellen

```

CREATE TABLE <NeueTabelle> AS SELECT * FROM <AlteTabelle>;
ALTER TABLE <NeueTabelle> ADD PRIMARY KEY (<Alter_Primary_Key>);

```

### 5.3.5 Erstellen von Sichten

```

CREATE VIEW <NeueSicht> AS SELECT <Spalten>
FROM <Tabelle>
WHERE <Bedingung>
WITH CHECK OPTION;

```

WITH CHECK OPTION → lehnt Änderungen der Sicht ab, wenn Zeilen verloren gehen würden

### 5.3.6 Löschen von Sichten

```

DROP VIEW <Sicht>;

```

### 5.3.7 Ändern von Sichten

- Löschen und neu erstellen

- aktualisieren mit `UPDATE` wie eine Tabelle, aber **nicht** möglich bei: Aggregatfunktionen, Verbund (mehrere Interpretationen), keine Schlüssel

## 5.4 Datentypen

- `INT` → Ganzzahl
- `FLOAT` → Gleitkommazahl
- `DECIMAL(i, j)` → Gleitkommazahl,  $i \rightarrow \#$  Stellen,  $j \rightarrow \#$  Nachkommastellen
- `CHAR(n)` → Zeichenkette der Länge  $n$
- `VARCHAR(n)` → Zeichenkette der maximalen Länge  $n$ 
  - Start und Ende markiert mit `'` (z.B. `'Hallo'`)
  - Konkatination mit `||` (z.B. `'Hallo' || 'Welt'`)
  - Mustervergleich mit `LIKE`, „%“ (beliebig viele Zeichen), „\_“ (ein einzelnes Zeichen) und `ESCAPE '<Zeichen>'` (Definition eines Escape-Zeichens)
- `DATE` → Datum
  - String zu Datum: `TO_DATE('<String>', '<Format>')` mit Format z.B. `YYYY-MM-DD`, `DD.MM.YYYY`
  - Datum zu Alter: `TRUNC(MONTHS_BETWEEN(SYSDATE, BirthDate)/12)`
  - Aktuellen Monat: `SELECT EXTRACT(MONTH FROM SYSDATE) FROM dual;`
- `TIME` → Uhrzeit
- `TIMESTAMP` → Datum und Uhrzeit

## 5.5 Privilegien

### 5.5.1 Vergabe

SQL

```
GRANT <Privilegienliste>
ON <Relation oder Sicht>
TO <Account-Liste>
WITH GRANT OPTION;
```

- Privilegien: `CREATE`, `ALTER`, `DROP` (unabhängig von Relation), `SELECT`, `INSERT`, `DELETE` und `UPDATE` (für eine Relation/Sicht), oder `ALL`
- `WITH GRANT OPTION` → erlaubt Weitergabe der Rechte an andere

### 5.5.2 Widerrufen

SQL

```
REVOKE <Privilegienliste>
ON <Relation oder Sicht>
FROM <Account-Liste>
CASCADE;
```

`CASCADE` → Widerruf der propagierten Rechte auch für andere, denen die Rechte vom Account weitergegeben wurden

Sonst: RESTRICT  $\rightarrow$  REVOKE schlägt fehl, sofern Rechte propagiert wurden

## 5.6 Rekursion

- Alle *indirekte*  $\rightarrow$  Transitive Hülle
- Transitive Hülle benötigt Rekursion  $\rightarrow$  nicht in **SQL-92**!
- Rekursion  $\rightarrow$  WITH RECURSIVE mit **SQL:1999**

Beispiel zu WITH RECURSIVE

## 5.7 Beispielaufgaben

|          |          |          |          |          |
|----------|----------|----------|----------|----------|
| W   F 24 | W   F 25 | W   F 26 | W   F 27 | W   F 28 |
| W   F 29 | W   F 30 | W   F 31 | W   F 32 | W   F 33 |
| W   F 34 | W   F 35 | W   F 36 | W   F 37 | W   F 38 |
| W   F 39 | W   F 40 | W   F 41 | W   F 42 | W   F 43 |
| W   F 44 | W   F 45 | W   F 46 | W   F 47 | W   F 48 |
| W   F 49 | W   F 50 | W   F 51 | W   F 52 | W   F 53 |



# Kapitel 6 Datenbanksprachen

## 6.1 Relationale Algebra

→ prozedurale Sprache (Wie berechnet man das Ergebnis)

### Bemerkung

Eingabe und Ergebnisse der Operationen sind Relationen → Operationen können geschachtelt werden!

Zwischenspeichern des Ergebnis:

$R_1 \leftarrow \text{Operation}(R_0)$

### 6.1.1 Grundoperationen

| Operation          | Notation                       | Beschreibung   |
|--------------------|--------------------------------|--|
| <b>Selektion</b>   | $\sigma_F(r)$                  | <ul style="list-style-type: none"><li>wählt <b>Zeilen</b>/Tupel aus Relation <math>r</math> aus, die Bedingung <math>F</math> erfüllen → <math>\sigma_F(r) := \{t \mid t \in r \wedge F(t) = \text{true}\}</math></li><li><math>F</math> ist Prädikat/Bedingung, bestehend aus Attributennamen, Konstanten, Vergleichsoperatoren <math>\{=, &lt;, &gt;, \leq, \geq, \neq\}</math> und logischen Verknüpfungen <math>\{\text{AND}, \text{OR}, \text{NOT}\}</math></li></ul> |
| <b>Projektion</b>  | $\pi_X(r)$                     | <ul style="list-style-type: none"><li>wählt <b>Spalten</b>/Attribute aus Relation <math>r</math> aus → <math>\pi_X(r) := \{t[X] \mid t \in r\}</math></li><li><math>X</math> ist Attributliste <math>(A_1, A_2, \dots, A_n)</math></li><li>filtert Duplikate heraus</li></ul>  |
| <b>Umbenennung</b> | $\rho_{S(B_1, \dots, B_n)}(r)$ | <ul style="list-style-type: none"><li><b>benennt</b> Attribute von Relation <math>r</math> und/oder Namen der Relation <b>um</b></li><li><math>B_1, \dots, B_n</math> sind neue Namen der Attribute, und <math>S</math> ist neuer Name der Relation</li><li>Auch möglich:<ul style="list-style-type: none"><li><math>\rho_{(B_1, \dots, B_n)}(r)</math> (nur Attributnamen)</li><li><math>\rho_S(r)</math> (nur Name der Relation)</li></ul></li></ul>                     |

### Mengenoperationen

|                    |            |   |
|--------------------|------------|---|
| <b>Vereinigung</b> | $r \cup s$ | $r \cup s := \{t \mid t \in r \vee t \in s\}$ für $r(R)$ und $s(S)$   |
| <b>Differenz</b>   | $r - s$    | $r - s := \{t \mid t \in r \wedge t \notin s\}$ für $r(R)$ und $s(S)$ |
| <b>Schnitt</b>     | $r \cap s$ | $r \cap s := \{t \mid t \in r \wedge t \in s\}$ für $r(R)$ und $s(S)$ |

|                             |                               |   |
|-----------------------------|-------------------------------|---|
|                             |                               | <ul style="list-style-type: none"> <li>nur wenn <b>vereinigungsverträglich!</b> (gleiche # Spalten, gleiche Wertebereiche)</li> <li>Attributennamen der ersten Relation werden übernommen</li> </ul>  |
| <b>Kartesisches Produkt</b> | $r \times s$                  | $r \times s := \{(t_1, t_2) \mid t_1 \in r \wedge t_2 \in s\}$ für $r(R)$ und $s(S)$<br><br>gleiche Attributennamen nicht erlaubt! (notfalls Umbenennung)   |
| <b>Allgemeiner Verbund</b>  | $r \bowtie_{\theta} s$        | <ul style="list-style-type: none"> <li><b>Verbund</b> mit Verbundbedingung <math>\theta</math> (wie Prädikat, keine Konstanten!)</li> <li>äquivalent zu <math>\sigma_{\theta}(r \times s)</math></li> <li>Tupel, deren Verbundattribut den Wert NULL hat, sind <b>nicht</b> im Ergebnis (<b>InnerJoin</b>)</li> <li>gleiche Attributennamen verboten! (notfalls Umbenennung)</li> </ul> |
| <b>Gleichheitsverbund</b>   | $r \bowtie_{\dots = \dots} s$ | Allgemeiner Verbund mit $\theta$ als =  |
| <b>Natürlicher Verbund</b>  | $r * s$                       | <ul style="list-style-type: none"> <li>Gleichheitsverbund, basierend auf <b>gleichen Attributnamen</b></li> <li>doppelte Attribute werden nur <b>einmal</b> im Ergebnis aufgeführt</li> </ul>   |
| <b>Äußerer Verbund</b>      |                               | <ul style="list-style-type: none"> <li>behält alle Tupel aus <math>r</math> oder/und aus <math>s</math> bei</li> <li>füllt nicht-übereinstimmende Tupel mit NULL auf</li> </ul>   |
|                             | $r \bowtie s$                 | <b>Left Outer Join</b> (alle Tupel aus $r$ )  |
|                             | $r \bowtie s$                 | <b>Right Outer Join</b> (alle Tupel aus $s$ )   |
|                             | $r \bowtie s$                 | <b>Full Outer Join</b> (alle Tupel aus $r$ und $s$ )  |

### 6.1.1.1 Beispiele

Selektion
Projektion
Umbenennung
Kreuzprodukt
Allg. Verbund  
Natürl. Verbund
Äußerer Verbund

### 6.1.2 Aggregatfunktionen

- Notation:  $\langle \text{Gruppierungsattribute} \rangle \mathcal{F}_{\langle \text{Aggregatfunktionen} \rangle}(r)$
- Aggregatfunktionen: COUNT, SUM, AVG, MIN, MAX
- Duplikate werden beim Aggregieren nicht entfernt
- Ergebnis: Relation mit allen Gruppierungsattributen und jeweils eine Spalte pro Aggregatfunktion

Beispiel zu 6.1.2 Aggregatfunktionen

### 6.1.3 Division

Für  $R(A, B)$  und  $S(B)$  ( $S \subseteq R$ ) ist die Division definiert als:

$$R \div S := \{a \in \pi_A(R) \mid \forall b \in S: (a, b) \in R\}$$

#### In einfachen Worten:

Die Ergebnisrelation  $E$  enthält alle Werte aus Spalte  $A$ , die mit jedem  $b$  aus  $S$  verknüpft, in  $R$  vorkommen. Es muss somit gelten:  $E \times S \subseteq R$ !

#### In Grundoperationen:

$$\begin{aligned} e(E) \leftarrow r(R) \div s(S) &\Leftrightarrow \begin{aligned} e_1 &\leftarrow \pi_E(r) \\ e_2 &\leftarrow \pi_E((s \times e_1) - r) \\ e &\leftarrow e_1 - e_2 \end{aligned} \end{aligned}$$

**Nutzen:** „Alle..., die an allen...“-Abfragen

#### Beispiel zu 6.1.3 Division

## 6.2 Relationaler Kalkül

→ deklarative (nicht prozedurale) Sprache (Was ist das Ergebnis)

### 6.2.1 Sichere Ausdrücke

- sicherer Ausdruck: endlich große Ergebnismenge
  - unsicherer Ausdruck: unendlich große Ergebnismenge → Berechnung endet nie
- Vermeidung von Endlosschleifen → nicht **berechnungsvollständig**

#### Beispiel unsicherer Ausdruck

### 6.2.2 Relationale Vollständigkeit

relational vollständig → Abfragesystem kann alle Operationen der relationalen Algebra umsetzen

Relationale Algebra und Relationaler Kalkül sind **relational äquivalent!** (gleichmächtig)

### 6.2.3 Unterschied Tupel- und Bereichskalkül

- **Tupelkalkül:** Formuliert Bedingungen über ganze Tupel, z. B.:

$$\{t \mid t \in Student \wedge t.Studiengang = 'Informatik'\}$$

Hierbei ist  $t$  ein Tupelvariable und die Bedingung bezieht sich auf Attribute des Tupels.

- **Bereichskalkül:** Formuliert Bedingungen direkt über die Attributwerte:

$$\{(s, n) \mid \exists m((s, n, m) \in Student \wedge m > 18)\}$$

Hier wird explizit auf einzelne Attribute referenziert, anstatt ganze Tupel zu verwenden.

### 6.2.4 Relationaler Tupelkalkül

$$\{t_1.A, t_2.B, \dots \mid \text{Bedingung}(t_1, t_2, \dots)\}$$

- $t_1, t_2, \dots$  sind Tupelvariablen
- $t_1.A, t_2.B, \dots$  sind die Ausgabeattribute
- Bedingung ist logischer Ausdruck über den Tupelvariablen, bestehend aus:
  - Vergleichsoperatoren ( $=, <, >, \leq, \geq, \neq$ )
  - logischen Operatoren (and, or, not)
  - Bereichsrelation  $R(t_i)$  ( $R$  Name einer Relation, quasi  $t_i \in R$ )
  - Quantoren  $(\exists t)(F(t))$  und  $(\forall t)(F(t))$  ( $F$  ist Bedingung)
    - Ausgabetupel sind implizit  $\exists$ -quantifiziert, alle anderen müssen explizit quantifiziert werden!

#### Logische Regeln

$$(\exists t)(F(t)) \equiv \neg(\forall t)(\neg F(t))$$

$$(\forall t)(F(t)) \equiv \neg(\exists t)(\neg F(t))$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$

$$A \rightarrow B \equiv \neg A \vee B$$

einfaches Beispiel

komplexeres Beispiel

### 6.3 Beispielaufgaben

W | F 54

W | F 55

W | F 56

W | F 57

W | F 58

W | F 59

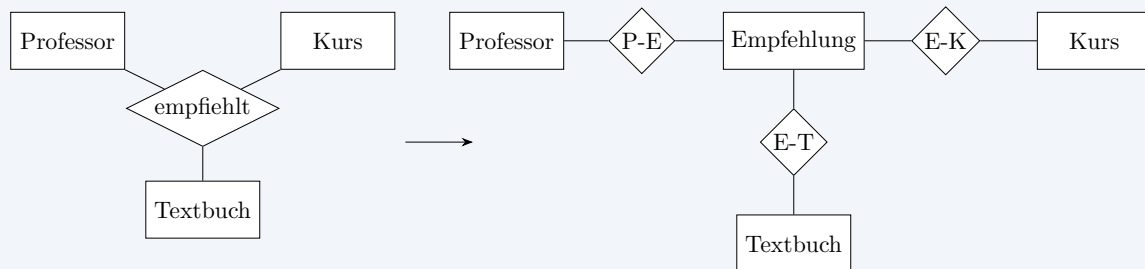
W | F 60

W | F 61

# Kapitel 7 Anhang

## 7.1 Beispiele

### Beispiel Zerlegung ternärer Beziehungsmenge



[Zum Text](#)

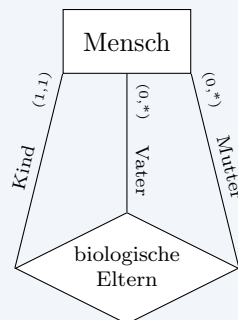
### Beispiel zu 2.3.1 Kardinalität



→ in Worten: „Ein Patient belegt genau 1 Zimmer und ein Zimmer kann von keinem bis zu beliebig vielen Patienten belegt werden.“

[Zum Text](#)

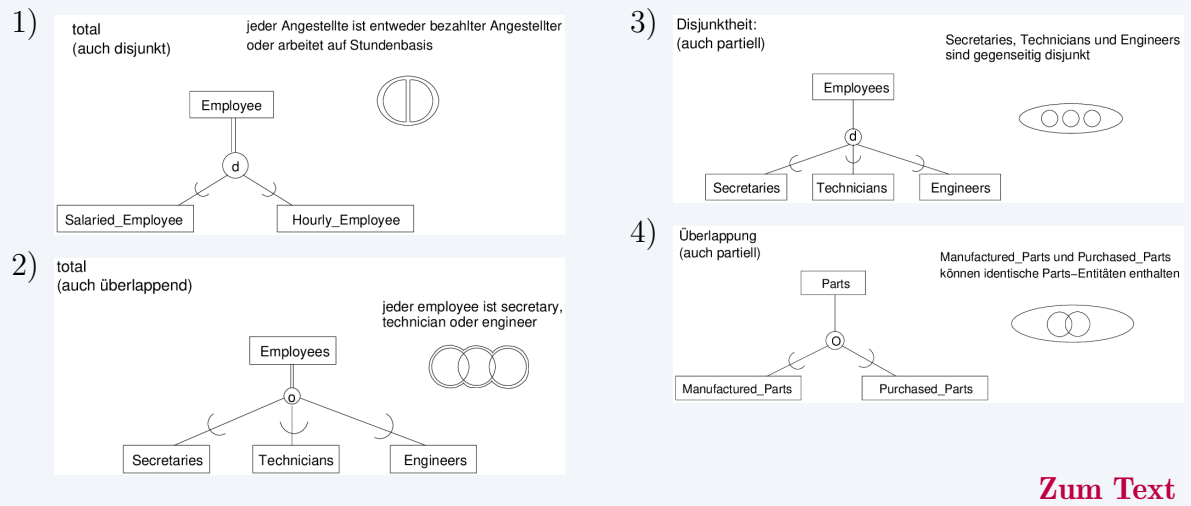
### Beispiel zu 2.3.2 Rekursive Beziehungsmenge



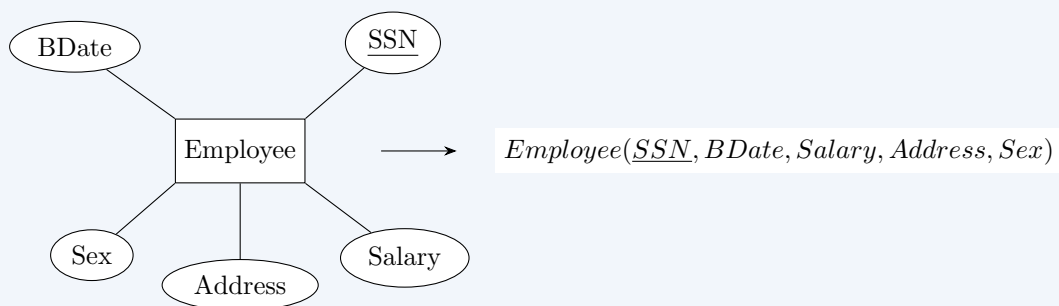
[Zum Text](#)

### Beispiel zu 2.5 Spezialisierung

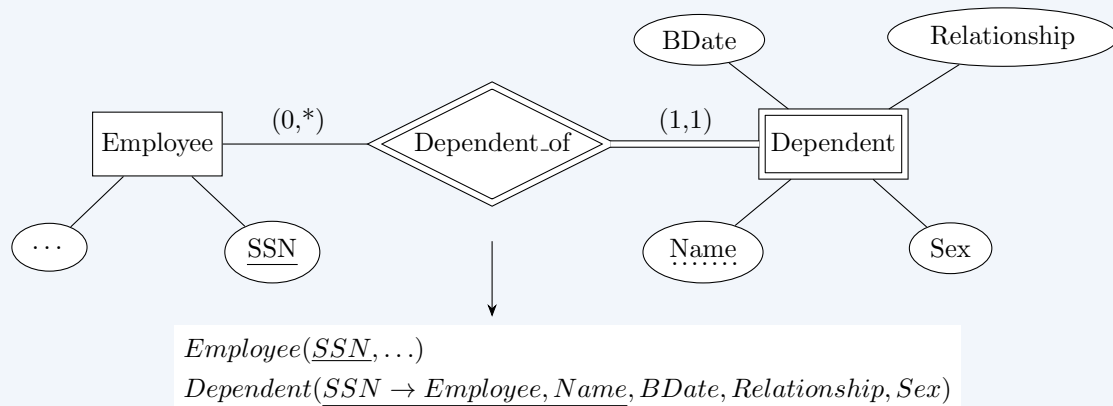
| Kombinationen: |          | disjunkt | überlappend |
|----------------|----------|----------|-------------|
|                | total    | 1)       | 2)          |
|                | partiell | 3)       | 4)          |



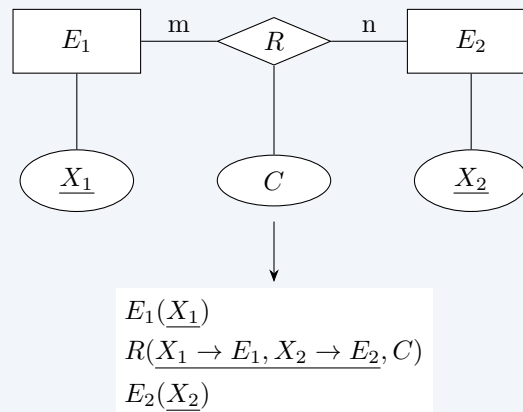
## Beispiel zu 3.2.1 Allgemeine Entitätenmenge



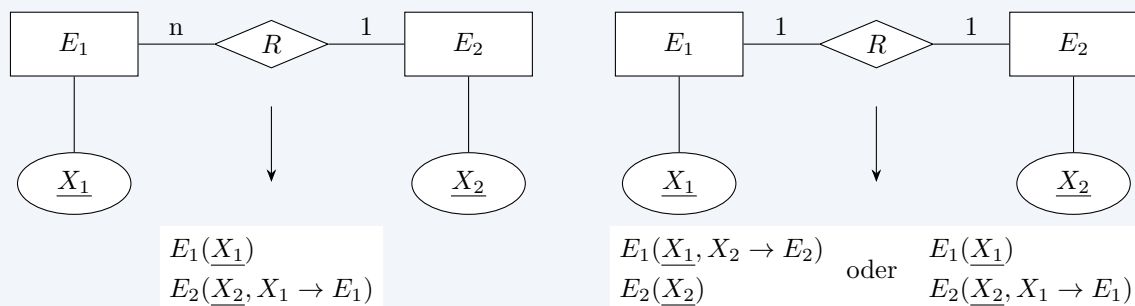
## Beispiel zu 3.2.2 Schwache Entitätenmenge



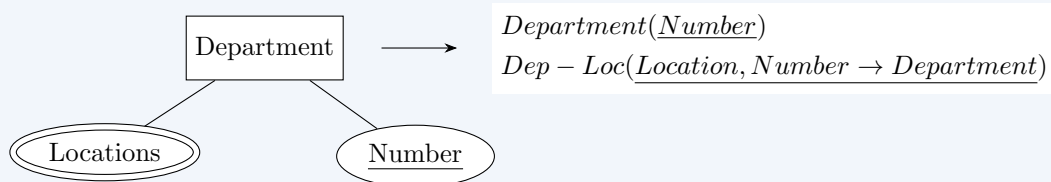
## Beispiel zu 3.2.3 Allgemeine Beziehungsmenge (m:n)

[Zum Text](#)

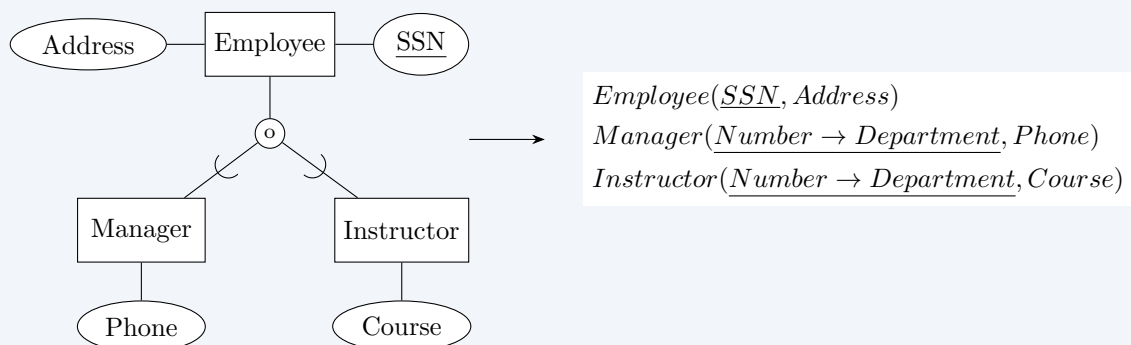
## Beispiel zu 3.2.4 Funktionale Abhängigkeiten (1:n, 1:1)

[Zum Text](#)

## Beispiel Transformation Mehrwertiges Attribut

[Zum Text](#)

## Beispiel zu 3.2.6 Spezialisierung

[Zum Text](#)

## weitere Optionen der Transformation

## Option 2

$Sub_i(\underline{X}, A_1, \dots, A_n, \dots)$  (jede Subklasse enthält die „generellen“ Attribute, keine Relation für die Superklasse)

## Option 3 (für disjunkt)

$Super(\underline{X}, A_1, \dots, A_n, \dots, t)$  (Attribut  $t$  bestimmt die Subklasse)

## Option 4 (für überlappend)

$Super(\underline{X}, A_1, \dots, A_n, \dots, t_1, \dots, t_m)$  („boolesches“ Attribut  $t_i$  gibt Zugehörigkeit des Tupel zur Subklasse  $S_i$  an)

[Zum Text](#)

## Beispiel Primärschlüssel

|                      |   |   |   |   |
|----------------------|---|---|---|---|
|                      | A   | B | C | D |
| Gegeben:             | 1   | a | 2 | x |
|                      | 2   | b | 2 | y |
|                      | 1   | c | 2 | z |
|                      | 3   | d | 2 | y |
|                      | 4   | e | 4 | y |
| Superschlüssel:      | {(AB), (AD), (ABC), (ABD),<br>(ACD), (B), (BC), (BD),<br>(BCD), (ABCD)} |   |   |   |
| Schlüsselkandidaten: | {(B), (AD)}   |   |   |   |
| Primärschlüssel:     | {(B)}   |   |   |   |

[Zum Text](#)

## Beispiel zu 4.3.2 Abschluss

$F = \{\{Ssn\} \rightarrow \{Ename, Bdate, Address, Dnumber\}, \{Dnumber\} \rightarrow \{Dname, Dmgrssn\}\}$

Folgende FDs bspw. sind in  $F^+$  enthalten (können aus  $F$  abgeleitet werden):

- $F \models \{Ssn\} \rightarrow \{Dname, Dmgrssn\}$
- $F \models \{Ssn\} \rightarrow \{Ssn\}$
- $F \models \{Dnumber\} \rightarrow \{Dname\}$

Und noch viele mehr...

[Zum Text](#)

## Beispiel zu 4.3.4 Attribut-Abschluss

Gegeben:  $F = \{\{Ssn\} \rightarrow \{Ename, Bdate, Address, Dnumber\}, \{Dnumber\} \rightarrow \{Dname, Dmgrssn\}\}$

Frage:  $\{Ssn\} \rightarrow \{Dmgrssn\} \in F^+?$



Lösung: Überprüfen von  $\{\text{Dmgrssn}\} \subseteq \{\text{Ssn}\}^+$  bzgl.  $F$

Berechnung von  $\{\text{Ssn}\}^+$  bzgl.  $F$ :

- Initialisierung:  $\{\text{Ssn}\}^+ = \{\text{Ssn}\}$
- Iteration 1:  
 $\{\text{Ssn}\} \rightarrow \{\text{Ename}, \text{Bdate}, \text{Address}, \text{Dnumber}\} \in F$  und  $\{\text{Ssn}\} \subseteq \{\text{Ssn}\}^+$   
 $\hookrightarrow \{\text{Ssn}\}^+ = \{\text{Ssn}, \text{Ename}, \text{Bdate}, \text{Address}, \text{Dnumber}\}$
- Iteration 2:  
 $\{\text{Dnumber}\} \rightarrow \{\text{Dname}, \text{Dmgrssn}\} \in F$  und  $\{\text{Dnumber}\} \subseteq \{\text{Ssn}\}^+$   
 $\hookrightarrow \{\text{Ssn}\}^+ = \{\text{Ssn}, \text{Ename}, \text{Bdate}, \text{Address}, \text{Dnumber}, \text{Dname}, \underline{\text{Dmgrssn}}\}$
- Iteration 3:  
keine Änderung  $\rightarrow$  Algorithmus fertig ✓

$\{\text{Dnumber}\} \subseteq \{\text{Ssn}\}^+ = \{\text{Ssn}, \text{Ename}, \text{Bdate}, \text{Address}, \text{Dnumber}, \text{Dname}, \underline{\text{Dmgrssn}}\}$ ,  
also gilt  $\{\text{Ssn}\} \rightarrow \{\text{Dmgrssn}\} \in F^+$ !

[Zum Text](#)

#### Beispiel zu 4.4 Mehrwertige Abhängigkeiten

MDV  $ISBN \twoheadrightarrow Autor$  ( $, ISBN \twoheadrightarrow Version$ ) auf  $r(ISBN, Autor, Version, Verlag)$

| $X$                                 | $Y$               | $R \setminus (X \cup Y)$ | Tupel                 |
|-------------------------------------|-------------------|--------------------------|-----------------------|
| ('978-3-86680-192-9')               | 'Hans Mustermann' | ('1', 'Klett')           | $t_1$                 |
| ('978-3-86680-192-9')               | 'Max Stemler'     | ('2', 'Klett')           | $t_2$                 |
| $\rightarrow$ ('978-3-86680-192-9') | 'Hans Mustermann' | ('2', 'Klett')           | $t_3$ muss existieren |
| $\rightarrow$ ('978-3-86680-192-9') | 'Max Stemler'     | ('1', 'Klett')           |                       |

[Zum Text](#)

#### Beispiel zu 4.5.1 Verbundtreue

| Erhalt   | Verletzung   |
|--|--|
| <ul style="list-style-type: none"> <li>• <math>R(A, B, C)</math> mit <math>F = \{A \rightarrow B, B \rightarrow C\}</math></li> <li>• Zerlegung in <math>R_1(A, B)</math> und <math>R_2(B, C)</math></li> <li>• <b>verlustfrei</b>:<br/> <math>R_1 \cap R_2 = B</math>, und <math>B \rightarrow BC \in F^+</math> ✓</li> </ul> | <ul style="list-style-type: none"> <li>• <math>R(A, B, C)</math> mit <math>F = \{A \rightarrow B, C \rightarrow B\}</math></li> <li>• Zerlegung in <math>R_1(A, B)</math> und <math>R_2(B, C)</math></li> <li>• <b>nicht verlustfrei</b>:<br/> <math>R_1 \cap R_2 = B</math>, aber <math>B \rightarrow AB \notin F^+</math> und<br/> <math>B \rightarrow BC \notin F^+</math> ✗</li> </ul> |

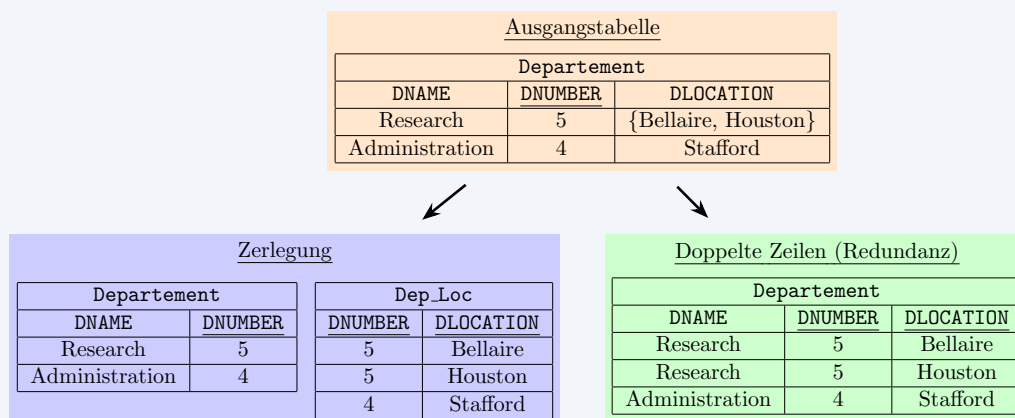
[Zum Text](#)

## Beispiel zu 4.5.2 Abhängigkeitsstreue

| Erhalt  | Verletzung  |
|---|---|
| <ul style="list-style-type: none"> <li><math>R(A, B, C, D)</math> mit <math>F = \{A \rightarrow BCD, D \rightarrow B\}</math></li> <li>Zerlegung in <math>R_1(A, B, D)</math> und <math>R_2(A, C)</math></li> <li><b>abhängigkeitsstreu:</b><br/> <math>\pi_{R_1}(F) = \{A \rightarrow BD, D \rightarrow B\}</math> und<br/> <math>\pi_{R_2}(F) = \{A \rightarrow C\}</math>, also<br/> <math>(\pi_{R_1}(F) \cup \pi_{R_2}(F))^+ = F^+ \checkmark</math></li> </ul> | <ul style="list-style-type: none"> <li><math>R(A, B, C, D)</math> mit <math>F = \{A \rightarrow BCD, D \rightarrow B\}</math></li> <li>Zerlegung in <math>R_1(A, B)</math> und <math>R_2(A, C, D)</math></li> <li><b>nicht abhängigkeitsstreu:</b><br/> <math>\pi_{R_1}(F) = \{A \rightarrow B\}</math> und<br/> <math>\pi_{R_2}(F) = \{A \rightarrow CD\}</math>, also<br/> <math>(\pi_{R_1}(F) \cup \pi_{R_2}(F))^+ \neq F^+</math>, da<br/> <math>D \rightarrow B \notin (\pi_{R_1}(F) \cup \pi_{R_2}(F))^+ \nless</math></li> </ul> |

Zum Text

## Beispiel zu 4.5.3 1. Normalform (1NF)



In beiden Fällen muss DLocation Teil des Schlüssel sein, um die Eindeutigkeit zu gewährleisten.

Zum Text

## Beispiel zu 4.5.4 2. Normalform (2NF)

1NF-Relationschema  $R(\underline{A}, B, C, D, E, F)$  mit  $F = \{B \rightarrow C, C \rightarrow D\}$   
 Zerlegung in 2NF:  $R_1(\underline{A}, \underline{B}, E, F)$  und  $R_2(\underline{B}, C, D)$

Zum Text

## Beispiel zu 4.5.5 3. Normalform (3NF)

2NF-Relationschema  $R(\underline{A}, B, C, D, E, F)$  mit  $F = \{B \rightarrow C, C \rightarrow D\}$   
 Zerlegung in 3NF:  $R_1(\underline{A}, B, E, F)$ ,  $R_2(\underline{B}, C)$  und  $R_3(\underline{C}, D)$

Zum Text

## Beispiel zu 4.5.6 Boyce-Codd-Normalform (BCNF)

3NF-Relationenschema  $R(\underline{A}, B, C)$  mit  $F = \{B \rightarrow A\}$ : bereits in BCNF ( $B \rightarrow A$  und  $A$  Schlüssel, bedeutet, dass  $B$  auch Schlüssel ist  $\checkmark$ )

Zum Text

## Beispiel zu 4.5.7 4. Normalform (4NF)

MVD  $ISBN \twoheadrightarrow Autor, ISBN \twoheadrightarrow Version, ISBN \twoheadrightarrow Verlagsname$

| <u>ISBN</u>   | <u>Titel</u>        | <u>Autor</u> | <u>Version</u> | <u>Verlagsname</u> |
|---------------|---------------------|--------------|----------------|--------------------|
| 0-201-03801-3 | Geographie Klasse 7 | Elmasri      | 3              | Klett              |
| 0-201-03801-3 | Geographie Klasse 7 | Navathe      | 3              | Klett              |
| 0-201-03801-3 | Geographie Klasse 7 | Elmasri      | 4              | Klett              |
| 0-201-03801-3 | Geographie Klasse 7 | Navathe      | 4              | Klett              |

Wird mittels Zerlegung in die 4NF gebracht:

| <u>ISBN</u>   | <u>Titel</u>        | <u>Verlagsname</u> |
|---------------|---------------------|--------------------|
| 0-201-03801-3 | Geographie Klasse 7 | Klett              |
| 0-201-03801-3 | Geographie Klasse 7 | Klett              |
| 0-201-03801-3 | Geographie Klasse 7 | Klett              |
| 0-201-03801-3 | Geographie Klasse 7 | Klett              |

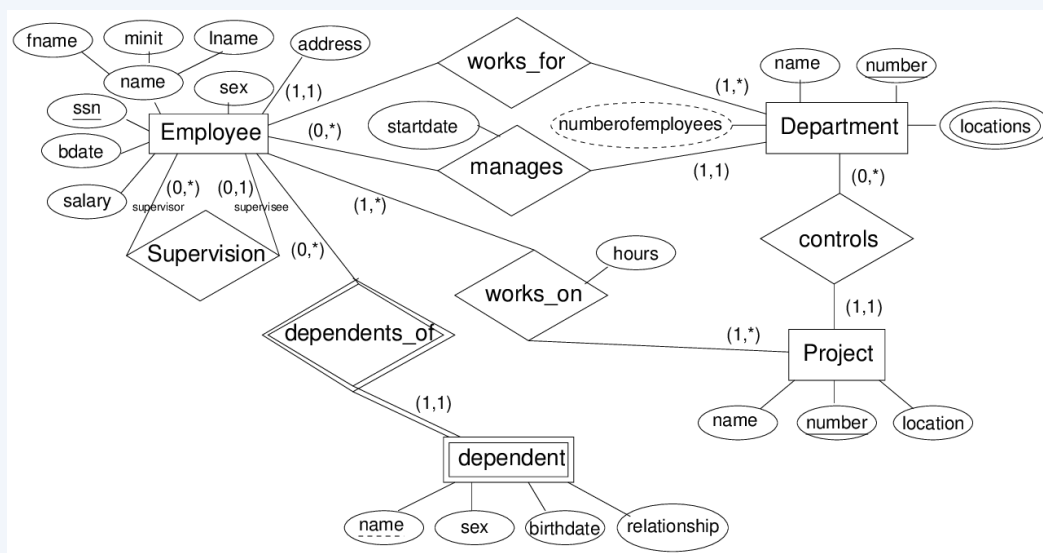
| <u>ISBN</u>   | <u>Autor</u> |
|---------------|--------------|
| 0-201-03801-3 | Elmasri      |
| 0-201-03801-3 | Navathe      |
| 0-201-03801-3 | Elmasri      |
| 0-201-03801-3 | Navathe      |

| <u>ISBN</u>   | <u>Version</u> |
|---------------|----------------|
| 0-201-03801-3 | 3              |
| 0-201-03801-3 | 3              |
| 0-201-03801-3 | 4              |
| 0-201-03801-3 | 4              |

[Zum Text](#)

## Beispieldatenbank aus der Vorlesung

ER-Modell:



Relationenschema:

- Employee(fname, minit, lname, ssn, bdate, address, sex, salary, superssn → Employee, dno → Department)
- Department(dname, dnumber, mgrssn → Employee, mgrstartdate)
- Dept\_locations(dnumber → Department, dlocation)
- Project(pname, pnumber, plocation, dnum → Department)
- Works\_on(essn → Employee, pno → Project, hours)
- Dependent(essn → Employee, dependent\_name, sex, bdate, relationship)

Daten:

| EMPLOYEE |       |         |           |            |                          |     |        |           |     |
|----------|-------|---------|-----------|------------|--------------------------|-----|--------|-----------|-----|
| FNAME    | MINIT | LNAME   | SSN       | BDATE      | ADDRESS                  | SEX | SALARY | SUPERSSN  | DNO |
| John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M   | 30000  | 333445555 | 5   |
| Franklin | T     | Wong    | 333445555 | 1955-12-08 | 638 Voss, Houston, TX    | M   | 40000  | 888665555 | 5   |
| Alicia   | J     | Zelaya  | 999887777 | 1968-07-19 | 3321 Castle, Spring, TX  | F   | 25000  | 987654321 | 4   |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX  | F   | 43000  | 888665555 | 4   |
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M   | 38000  | 333445555 | 5   |
| Joyce    | A     | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX   | F   | 25000  | 333445555 | 5   |
| Ahmad    | V     | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX  | M   | 25000  | 987654321 | 4   |
| James    | E     | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston, TX   | M   | 55000  | null      | 1   |

| DEPARTMENT     |         |           |              |
|----------------|---------|-----------|--------------|
| DNAME          | DNUMBER | MGRSSN    | MGRSTARTDATE |
| Research       | 5       | 333445555 | 1988-05-22   |
| Administration | 4       | 987654321 | 1995-01-01   |
| Headquarters   | 1       | 888665555 | 1981-06-19   |

| DEPT LOCATIONS |           |
|----------------|-----------|
| DNUMBER        | DLOCATION |
| 1              | Houston   |
| 4              | Stafford  |
| 5              | Bellaire  |
| 5              | Sugarland |
| 5              | Houston   |

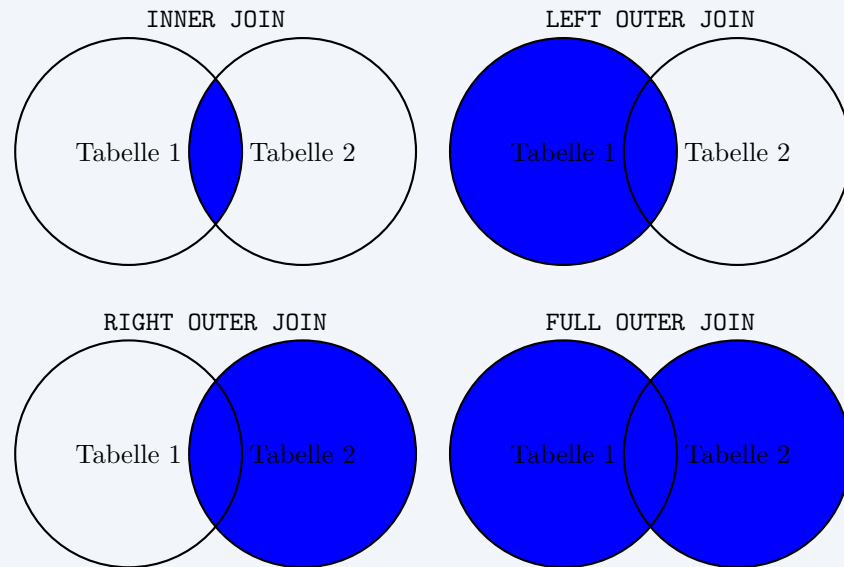
| DEPENDENT |                |     |            |              |
|-----------|----------------|-----|------------|--------------|
| ESSN      | DEPENDENT_NAME | SEX | BDATE      | RELATIONSHIP |
| 333445555 | Alice          | F   | 1986-04-05 | DAUGHTER     |
| 333445555 | Theodore       | M   | 1983-10-25 | SON          |
| 333445555 | Joy            | F   | 1958-05-03 | SPOUSE       |
| 987654321 | Abner          | M   | 1942-02-28 | SPOUSE       |
| 123456789 | Michael        | M   | 1988-01-04 | SON          |
| 123456789 | Alice          | F   | 1988-12-30 | DAUGHTER     |
| 123456789 | Elizabeth      | F   | 1967-05-05 | SPOUSE       |

| PROJECT         |         |           |      |
|-----------------|---------|-----------|------|
| PNAME           | PNUMBER | PLOCATION | DNUM |
| ProductX        | 1       | Bellaire  | 5    |
| ProductY        | 2       | Sugarland | 5    |
| ProductZ        | 3       | Houston   | 5    |
| Computerization | 10      | Stafford  | 4    |
| Reorganization  | 20      | Houston   | 1    |
| Newbenefits     | 30      | Stafford  | 4    |

| WORKS ON  |     |       |
|-----------|-----|-------|
| ESSN      | PNO | HOURS |
| 123456789 | 1   | 32,5  |
| 123456789 | 2   | 7,5   |
| 666884444 | 3   | 40,0  |
| 453453453 | 1   | 20,0  |
| 453453453 | 2   | 20,0  |
| 333445555 | 2   | 10,0  |
| 333445555 | 3   | 10,0  |
| 333445555 | 10  | 10,0  |
| 333445555 | 20  | 10,0  |
| 999887777 | 30  | 30,0  |
| 999887777 | 10  | 10,0  |
| 987987987 | 10  | 35,0  |
| 987987987 | 30  | 5,0   |
| 987654321 | 30  | 20,0  |
| 987654321 | 20  | 15,0  |
| 888665555 | 20  | null  |

SQL-Befehle für diese Datenbank siehe [VL\\_Beispiel.sql](#)[Zum Text](#)

## Bildliche Darstellung



Konkrete Beispiele siehe [hier](#)

[Zum Text](#)

## Beispiel zu WITH RECURSIVE

| Verbindung   |              |
|--------------|--------------|
| Von          | Nach         |
| Magdeburg    | Braunschweig |
| Braunschweig | Kassel       |
| Kassel       | Frankfurt    |

Welche Städte sind von Magdeburg aus (Umsteigen) erreichbar?

SQL

```

with recursive Erreichbar (von, nach) as (
  (SELECT von, nach
   FROM Verbindung
   WHERE von = 'Magdeburg')
  UNION ALL
  (SELECT v.von, n.nach
   FROM Erreichbar v, Verbindung n
   WHERE v.nach = n.von)
)
SELECT nach FROM Erreichbar;
```

[Zum Text](#)

## Selektion

- Wähle alle Weinkeller-Einträge (CELLAR) aus, bei denen READY=2000 ist:  
 $\sigma_{\text{READY}=2000}(\text{CELLAR})$
- Wähle alle Angestellten von Abteilung 4 mit einem Gehalt größer als 25000 oder von Abteilung 5 mit einem Gehalt größer als 30000:  
 $\sigma_{\text{DEPTNO}=4 \text{ AND SALARY}>25000} \text{ OR } (\text{DEPTNO}=5 \text{ AND SALARY}>30000)(\text{EMPLOYEE})$

|   |        |     |     |   |            |     |     |
|---|--------|-----|-----|---|------------|-----|-----|
| • | $r(R)$ |     |     | und $r_1 \leftarrow \sigma_{(A>2 \text{ AND } (\text{NOT } B<5))}(r)$ ergibt: | $r_1(R_1)$ |     |     |
|   | $A$    | $B$ | $C$ |   | $A$        | $B$ | $C$ |
|   | 1      | 2   | 3   |   | 4          | 5   | 6   |
|   | 4      | 5   | 6   |   |            |     |     |
|   | 2      | 5   | 3   |   |            |     |     |

[Zum Text](#)

## Projektion

- Wähle LNAME, FNAME und SALARY von EMPLOYEE:  
 $\pi_{(LNAME,FNAME,SALARY)}(\text{EMPLOYEE})$
- Wähle BIN# und WINE von CELLAR:  $\pi_{(BIN\#,WINE)}(\text{CELLAR})$

|   |        |     |     |   |            |     |                              |
|---|--------|-----|-----|---|------------|-----|------------------------------|
| • | $r(R)$ |     |     | und $r_1 \leftarrow \pi_{(B,C)}(r)$ ergibt: | $r_1(R_1)$ |     | (Beseitigung von Duplikaten) |
|   | $A$    | $B$ | $C$ |   | $B$        | $C$ |                              |
|   | 1      | 5   | 4   |   | 5          | 4   |                              |
|   | 2      | 5   | 2   |   | 5          | 2   |                              |
|   | 3      | 5   | 2   |   | 6          | 3   |                              |

[Zum Text](#)

## Umbenennung

|        |     |     |   |        |     |     |
|--------|-----|-----|---|--------|-----|-----|
| $r(R)$ |     |     | und $s \leftarrow \rho_{S(X,Y,Z)}(r)$ ergibt: | $s(S)$ |     |     |
| $A$    | $B$ | $C$ |   | $X$    | $Y$ | $Z$ |
| 1      | 5   | 4   |   | 1      | 5   | 4   |
| 2      | 5   | 2   |   | 2      | 5   | 2   |
| 3      | 5   | 2   |   | 3      | 5   | 2   |

[Zum Text](#)

## Kreuzprodukt

|        |     |          |                                       |        |     |     |     |
|--------|-----|----------|---------------------------------------|--------|-----|-----|-----|
| $r(R)$ |     | , $s(S)$ | und $t \leftarrow r \times s$ ergibt: | $t(T)$ |     |     |     |
| $A$    | $B$ |          |                                       | $A$    | $B$ | $C$ | $D$ |
| 1      | 2   |          |                                       | 1      | 2   | 5   | 7   |
| 3      | 4   |          |                                       | 1      | 2   | 8   | 3   |
|        |     |          |                                       | 3      | 4   | 5   | 7   |

[Zum Text](#)

## Allg. Verbund

|        |      |          |  |        |     |     |     |
|--------|------|----------|--|--------|-----|-----|-----|
| $r(R)$ |      | , $s(S)$ | und $t \leftarrow r \bowtie_{A<C} s$ ergibt: | $t(T)$ |     |     |     |
| $A$    | $B$  |          |  | $A$    | $B$ | $C$ | $D$ |
| 1      | 2    |          |  | 3      | 4   | 2   | 5   |
| 3      | 4    |          |  |        |     |     |     |
| 0      | NULL |          |  |        |     |     |     |

[Zum Text](#)

## Natürl. Verbund

| $r(R)$ |      |   | $s(S)$ |     |  | $t(T)$ |     |     |
|--------|------|---|--------|-----|--|--------|-----|-----|
| $A$    | $B$  |   | $C$    | $D$ |  | $A$    | $B$ | $D$ |
| 1      | 2    | , | 2      | 5   | und $t \leftarrow r * (\rho_{B,D}(s))$ ergibt: | 1      | 2   | 5   |
| 3      | 4    |   | 4      | 6   |  | 3      | 4   | 6   |
| 0      | NULL |   |        |     |  |        |     |     |

[Zum Text](#)

## Äußerer Verbund

linker äußerer Verbund:

| $r(R)$ |      | $s(S)$ |      | $r \bowtie_{B=C} s$ |      |      |      |
|--------|------|--------|------|---------------------|------|------|------|
| $A$    | $B$  | $C$    | $D$  | $A$                 | $B$  | $C$  | $D$  |
| 1      | 2    | 2      | 1    | 1                   | 2    | 2    | 1    |
| 3      | 4    | 4      | 2    | 3                   | 4    | 4    | 2    |
| 5      | 6    | 7      | 3    | 5                   | 6    | NULL | NULL |
| 2      | NULL | 8      | NULL | 2                   | NULL | NULL | NULL |

rechter äußerer Verbund:

| $r(R)$ |      | $s(S)$ |      | $r \bowtie_{B=C} s$ |      |     |      |
|--------|------|--------|------|---------------------|------|-----|------|
| $A$    | $B$  | $C$    | $D$  | $A$                 | $B$  | $C$ | $D$  |
| 1      | 2    | 2      | 1    | 1                   | 2    | 2   | 1    |
| 3      | 4    | 4      | 2    | 3                   | 4    | 4   | 2    |
| 5      | 6    | 7      | 3    | NULL                | NULL | 7   | 3    |
| 2      | NULL | 8      | NULL | NULL                | NULL | 8   | NULL |

voller äußerer Verbund:

| $r(R)$ |      | $s(S)$ |      | $r \bowtie_{B=C} s$ |      |      |      |
|--------|------|--------|------|---------------------|------|------|------|
| $A$    | $B$  | $C$    | $D$  | $A$                 | $B$  | $C$  | $D$  |
| 1      | 2    | 2      | 1    | 1                   | 2    | 2    | 1    |
| 3      | 4    | 4      | 2    | 3                   | 4    | 4    | 2    |
| 5      | 6    | 7      | 3    | 5                   | 6    | NULL | NULL |
| 2      | NULL | 8      | NULL | 2                   | NULL | NULL | NULL |
|        |      |        |      | NULL                | NULL | 7    | 3    |
|        |      |        |      | NULL                | NULL | 8    | NULL |

[Zum Text](#)

## Beispiel zu 6.1.2 Aggregatfunktionen

| $r(R)$ |     |     |   | $r_1(R_1)$ |  |
|--------|-----|-----|---|------------|--|
| $A$    | $B$ | $C$ |   | AVERAGE_C  |  |
| 1      | 1   | 5   | • und $r_1 \leftarrow \mathcal{F}_{\text{AVERAGE } C}(r)$ ergibt: | 4.2        |  |
| 1      | 2   | 6   |   |            |  |
| 2      | 3   | 2   |   |            |  |
| 2      | 3   | 5   |   |            |  |
| 3      | 4   | 3   |   |            |  |

- $r \leftarrow \text{DNO} \mathcal{F}_{\text{COUNT } SSN, \text{AVERAGE } SALARY}(\text{EMPLOYEE})$  ergibt:

| $r(R)$ |           |                |
|--------|-----------|----------------|
| DNO    | COUNT_SSN | AVERAGE_SALARY |
| 5      | 4         | 33250          |
| 4      | 3         | 31000          |
| 1      | 1         | 55000          |

[Zum Text](#)

## Beispiel zu 6.1.3 Division

Studenten mit besuchten Kursen (R):

| Student | Kurs   |
|---------|--------|
| Alice   | Mathe  |
| Alice   | Physik |
| Bob     | Mathe  |
| Bob     | Physik |
| Bob     | Chemie |
| Carol   | Mathe  |
| Carol   | Physik |
| Carol   | Chemie |

Alle angebotenen Kurse (S):

| Kurs   |
|--------|
| Mathe  |
| Physik |
| Chemie |

Fragestellung: „Welche Studenten haben alle Kurse besucht?“

Ergebnis der Division  $E = R \div S$ :

| Student |
|---------|
| Bob     |
| Carol   |

(Alice fehlt Chemie)

[Zum Text](#)

## Beispiel unsicherer Ausdruck

Ein klassisches Beispiel für einen unsicheren Ausdruck im relationalen Tupelkalkül ist:

$$\{t \mid \neg(t \in R)\}$$

Dieser Ausdruck beschreibt die Menge aller Tupel, die **nicht** in der Relation  $R$  enthalten sind. Da das Universum aller möglichen Tupel potenziell unendlich ist, könnte diese Abfrage eine unendliche Ergebnismenge liefern und ist daher unsicher.

[Zum Text](#)

## einfaches Beispiel

Finde Vor- und Nachnamen der Angestellten, deren Gehalt größer als \$5000:

$$\{t.FNAME, t.LNAME \mid \text{EMPLOYEE}(t) \text{ and } t.SALARY > 5000\}$$

[Zum Text](#)



## komplexeres Beispiel

Suche für alle Angestellten den Vor- und Nachnamen und den Vor- und Nachnamen des direkten Vorgesetzten:  
 $\{e.FNAME, e.LNAME, s.FNAME, s.LNAME \mid \text{EMPLOYEE}(e) \text{ and EMPLOYEE}(s) \text{ and } e.SUPERSSN=s.SSN\}$

[Zum Text](#)

## 7.2 Beweise

## Beweis der Gültigkeit der abgeleiteten Regeln

Abgeleitete Regeln:

- Vereinigung:  $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$

$$\begin{aligned} F^+ &= \{X \rightarrow Y, X \rightarrow Z\} \\ &= \{X \rightarrow Y, X \rightarrow Z, \underline{X \rightarrow XZ}, XZ \rightarrow YZ\} \quad (\text{Erweiterung}) \\ &= \{X \rightarrow Y, X \rightarrow Z, X \rightarrow XZ, XZ \rightarrow YZ, \underline{\underline{X \rightarrow YZ}}\} \quad (\text{Transitivität}) \end{aligned}$$

- Zerlegung:  $\{X \rightarrow YZ\} \models X \rightarrow Y$

$$\begin{aligned} F^+ &= \{X \rightarrow YZ\} \\ &= \{X \rightarrow YZ, \underline{YZ \rightarrow Y}\} \quad (\text{Reflexivität}) \\ &= \{X \rightarrow YZ, YZ \rightarrow Y, \underline{\underline{X \rightarrow Y}}\} \quad (\text{Transitivität}) \end{aligned}$$

- Pseudotransitivität:  $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$

$$\begin{aligned} F^+ &= \{X \rightarrow Y, WY \rightarrow Z\} \\ &= \{X \rightarrow Y, WY \rightarrow Z, \underline{WX \rightarrow WY}\} \quad (\text{Erweiterung}) \\ &= \{X \rightarrow Y, WY \rightarrow Z, WX \rightarrow WY, \underline{\underline{WX \rightarrow Z}}\} \quad (\text{Transitivität}) \end{aligned}$$

□

[Zum Text](#)

## Beweis, dass BCNF nicht immer abhängigkeitstreu realisierbar ist

$R(\underline{A}, B, C)$  mit  $F = \{C \rightarrow B\}$

Die drei möglichen Zerlegungen:

- 1)  $R_1(\underline{A}, B)$  und  $R_2(\underline{A}, C)$  (nicht verbundtreu)
- 2)  $R_1(\underline{A}, B)$  und  $R_2(\underline{B}, C)$  (nicht verbundtreu)
- 3)  $R_1(\underline{A}, C)$  und  $R_2(\underline{C}, B)$

Zerlegung 3 ist als einzige verbundtreu:

$R_1 \cap R_2 = C$ , und  $C \rightarrow BC \in F^+ \checkmark$  ( $C \rightarrow B$  und mit  $C$  erweitert)

aber nicht abhängigkeitsstreu:

$$\begin{aligned}
 \pi_{R_1}(F) &= \{AC \rightarrow AC, AC \rightarrow A, AC \rightarrow C\} \\
 \pi_{R_2}(F) &= \{C \rightarrow B, C \rightarrow BC\} \\
 \rightarrow (\pi_{R_1}(F) \cup \pi_{R_2}(F))^+ &= \underbrace{\pi_{R_1}(F) \cup \pi_{R_2}(F)}_{\text{„bisher“}} \cup \underbrace{\{AC \rightarrow B, AC \rightarrow BC\}}_{\text{„neu abgeleitet“}} \\
 AB \rightarrow C &\notin (\pi_{R_1}(F) \cup \pi_{R_2}(F))^+ \not\subseteq (F^+) \\
 \implies F^+ &\neq (\pi_{R_1}(F) \cup \pi_{R_2}(F))^+
 \end{aligned}$$

□

[Zum Text](#)

## 7.3 „Wahr oder Falsch“-Fragen

W | F 1

In einer Datenbank sollen die Kneipenvorlieben von Studierenden erfasst werden. Jede Kneipe hat einen Namen, einen Inhaber sowie PLZ, Ort und Straße. Zu den Studierenden werden Matrikelnummer, Name, Vorname und der Studiengang verwaltet. Jeder Student bevorzugt mindestens eine Kneipe und eine Kneipe kann von beliebig vielen Studenten bevorzugt werden.

Erstellen Sie ein ER-Diagramm für diesen Sachverhalt und erläutern Sie Ihre Modellierungsentscheidungen.

[Zur Lösung](#) [Zum Text](#)

W | F 2

Erstellen Sie ein ER-Diagramm zu folgendem Sachverhalt:

*Es soll eine Datenbank zur Verwaltung von Mannschaften und Spielen bei einem internationalen Sportturnier erstellt werden. Jede Mannschaft wird eindeutig durch das Land identifiziert, aus dem sie kommt, und besitzt einen Trainer. Jeder Spieler ist genau einer Mannschaft zugeordnet und wird durch eine fortlaufende Trikotnummer innerhalb dieser Mannschaft eindeutig identifiziert. Zusätzlich sollen der Name, das Geburtsdatum und das Alter eines Spielers abgefragt werden können. Zu jedem Spiel zwischen zwei Mannschaften wird das Datum, sowie die Anzahl der Tore der ersten und der zweiten Mannschaft gespeichert. Erstellen Sie ein ER-Diagramm für diesen Sachverhalt.*

[Zur Lösung](#) [Zum Text](#)

W | F 3

Erstellen Sie ein ER-Diagramm zu folgendem Sachverhalt:

*Für einen Hersteller von Kameras soll eine Datenbank entworfen werden. Dazu wird für jede hergestellte Kamera eine eindeutige Seriennummer gespeichert. Jeder Kunde, identifiziert durch eine eindeutige Kundennummer, besitzt einen Namen. Beinhaltet ein Kauf eines Kunden eine oder mehrere Kameras, werden diese zu einem Einkauf zusammengefasst und dieser mit einem Kaufdatum und einer eindeutigen Rechnungsnummer abgespeichert. Jede Kamera kann genau einmal gekauft werden. Im Fall einer Reklamation bzgl. eines Einkaufs sind jeweils Reklamationsdatum, Problembeschreibung und eine eindeutige Reklamationsnummer abzulegen. Ein Einkauf kann mehrfach*

reklamiert werden.

[Zur Lösung](#) [Zum Text](#)

#### W | F 4

Erstellen Sie ein ER-Diagramm zu folgendem Sachverhalt:

*Eine Universität hat einen eindeutigen Namen und besteht aus Gebäuden, die universitätsübergreifend eindeutig durch eine Nummer gekennzeichnet sind. Jedes Gebäude gehört zu genau einer Universität. Jedes Gebäude hat mindestens ein Zimmer und alle Zimmer sind fortlaufend innerhalb der Gebäude nummeriert. In jedem Zimmer arbeitet ein Mitarbeiter.*

[Zur Lösung](#) [Zum Text](#)

#### W | F 5

Erstellen Sie ein ER-Diagramm zu folgendem Sachverhalt:

*Es sollen die Daten über Rechnungen gespeichert werden. Jede Rechnung enthält eine eindeutige Nummer, einen Verkäufer, ein Datum und einen Gesamtpreis. Eine Rechnung beinhaltet weiterhin eine Anzahl von Rechnungsposten. Die Anzahl kann natürlich variieren. Jeder Posten hat zur Identifizierung innerhalb einer Rechnung eine laufende Nummer. Weiterhin hat er eine Produktbezeichnung, eine Anzahl der Produkte und einen Einzelpreis. Jeder Rechnungsposten ist genau einer Rechnung zugeordnet. Keine Rechnung kommt ohne Posten aus. Der Gesamtpreis einer Rechnung ergibt sich aus der Summe der entsprechenden Postenpreise, die sich jeweils aus dem Produkt aus Einzelpreis und Anzahl errechnen.*

[Zur Lösung](#) [Zum Text](#)

#### W | F 6

Erstellen Sie ein ER-Diagramm zu folgendem Sachverhalt. Nutzen Sie Spezialisierungen!

*Eine Universität hat verschiedene Angestellte, die alle entweder im akademischen oder nicht-akademischen Bereich arbeiten. Zum akademischen Personal zählen nur Professoren und wissenschaftliche Mitarbeiter. Alle wissenschaftlichen Mitarbeiter sind Haushaltsmitarbeiter, Drittmittelmitarbeiter oder beides gleichzeitig. Im nicht-akademischen Bereich arbeiten Sekretärinnen, Handwerker, Labor-Ingenieure und weitere nicht-akademische Angestellte.*

[Zur Lösung](#) [Zum Text](#)

#### W | F 7

Erstellen Sie ein ER-Diagramm zu folgendem Sachverhalt:

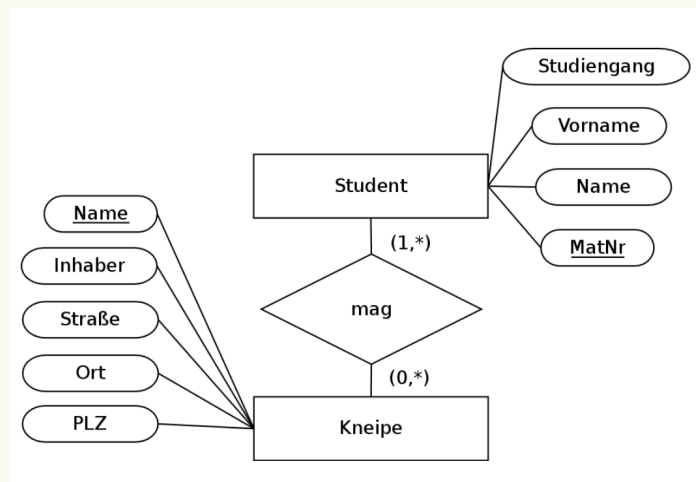
*Es soll eine Datenbank zur Krankenhausverwaltung erstellt werden. Ein Krankenhaus (das Krankenhaus selbst müssen Sie nicht modellieren!) besteht aus Stationen. Jede Station besitzt ein bestimmtes Aufgabengebiet und eine eindeutige Stationsnummer. Eine Station teilt sich in beliebig viele Zimmer auf, wobei für jedes Zimmer die Anzahl der verfügbaren Betten und zur Identifikation eine innerhalb der Station fortlaufende Zimmernummer gespeichert wird. Jeder Mitarbeiter des Krankenhauses arbeitet auf genau einer Station und verfügt jeweils über einen Namen und eine eindeutige Personalnummer. Ein Mitarbeiter ist entweder Arzt oder Pflegepersonal, weitere Arten von Mitarbeitern werden nicht in die Datenbank aufgenommen. Ärzte besitzen einen Rang und ein Fachgebiet und sind für die Behandlung von Patienten zuständig. Ein Patient*

bekommt bei seiner Einweisung eine eindeutige Patientenummer, besitzt einen Namen und eine bestimmte Krankheit. Jeder Patient belegt genau ein Zimmer, wobei sich mehrere Patienten ein Zimmer teilen können. Weiterhin wird zu jeder Zimmerbelegung das Anfangs- und das Enddatum gespeichert.

[Zur Lösung](#) [Zum Text](#)

### W | F 8

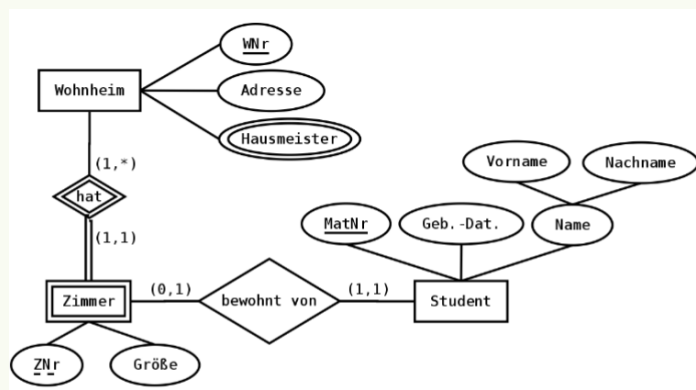
Das folgende ER-Diagramm soll möglichst semantikerhaltend in das Relationenmodell überführt werden. Verwenden Sie die textuelle Notation, um die entstehenden Relationenschemata anzugeben. Vergessen Sie die Angabe der Primär- und Fremdschlüssel nicht!



[Zur Lösung](#) [Zum Text](#)

### W | F 9

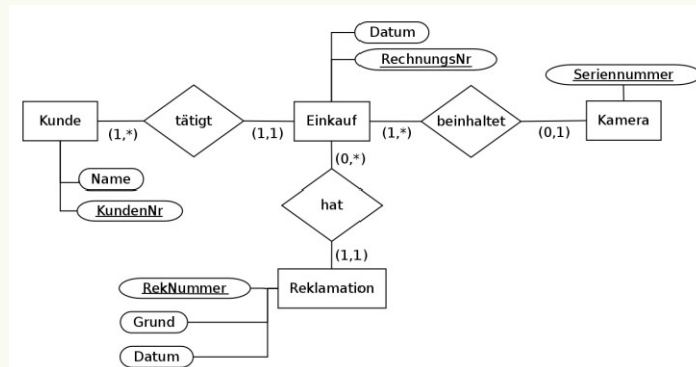
Das folgende ER-Diagramm soll möglichst semantikerhaltend in das Relationenmodell überführt werden. Verwenden Sie die textuelle Notation, um die entstehenden Relationenschemata anzugeben. Vergessen Sie die Angabe der Primär- und Fremdschlüssel nicht!



[Zur Lösung](#) [Zum Text](#)

### W | F 10

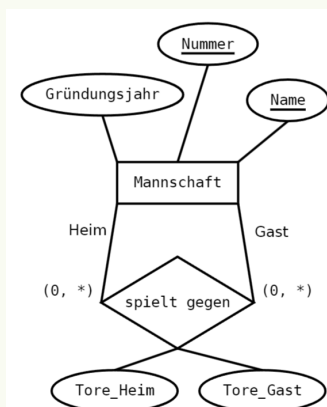
Das folgende ER-Diagramm soll möglichst semantikerhaltend in das Relationenmodell überführt werden. Verwenden Sie die textuelle Notation, um die entstehenden Relationenschemata anzugeben. Vergessen Sie die Angabe der Primär- und Fremdschlüssel nicht!



[Zur Lösung](#) [Zum Text](#)

W | F 11

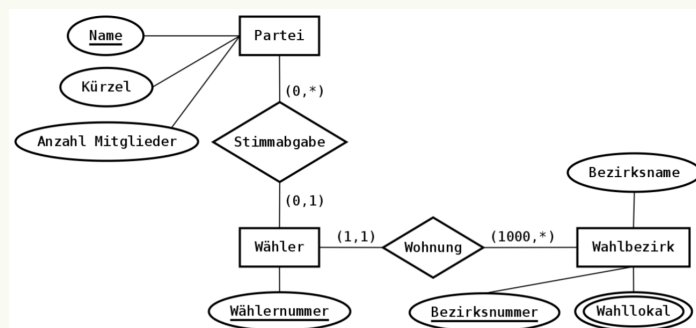
Das folgende ER-Diagramm soll möglichst semantikerhaltend in das Relationenmodell überführt werden. Verwenden Sie die textuelle Notation, um die entstehenden Relationenschemata anzugeben. Vergessen Sie die Angabe der Primär- und Fremdschlüssel nicht!



[Zur Lösung](#) [Zum Text](#)

W | F 12

Das folgende ER-Diagramm soll möglichst semantikerhaltend in das Relationenmodell überführt werden. Verwenden Sie die textuelle Notation, um die entstehenden Relationenschemata anzugeben. Vergessen Sie die Angabe der Primär- und Fremdschlüssel nicht!

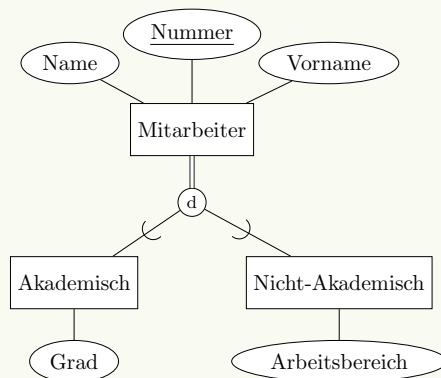


[Zur Lösung](#) [Zum Text](#)

W | F 13

Das folgende ER-Diagramm soll möglichst semantikerhaltend in das Relationenmodell überführt werden. Verwenden Sie die textuelle Notation, um die entstehenden Relationenschemata anzugeben. Nutzen Sie alle 4 möglichen Transformationen von einer

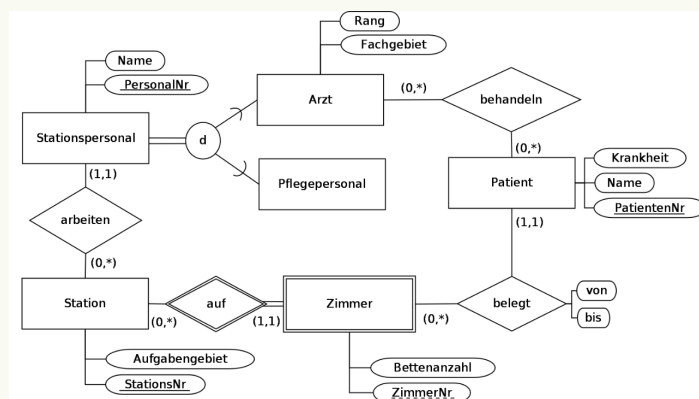
Spezialisierung. Vergessen Sie die Angabe der Primär- und Fremdschlüssel nicht!



[Zur Lösung](#) [Zum Text](#)

W | F 14

Das folgende ER-Diagramm soll möglichst semantikerhaltend in das Relationenmodell überführt werden. Verwenden Sie die textuelle Notation, um die entstehenden Relationenschemata anzugeben. Vergessen Sie die Angabe der Primär- und Fremdschlüssel nicht!



[Zur Lösung](#) [Zum Text](#)

W | F 15

Gegeben ist die folgende Relation:

| A | B | C | D | E |
|---|---|---|---|---|
| 1 | 2 | 5 | 1 | 4 |
| 2 | 3 | 5 | 2 | 1 |
| 2 | 3 | 4 | 3 | 1 |
| 3 | 3 | 4 | 3 | 3 |

Verletzt diese Relation folgende funktionale Abhängigkeiten?

$$F = \{A \rightarrow B, B \rightarrow A, AB \rightarrow A, CD \rightarrow AB, ABCDE \rightarrow ABCDE, ABE \rightarrow ABD\}$$

[Zur Lösung](#) [Zum Text](#)

W | F 16

Berechnen Sie die Menge  $F^+$  für das Relationenschema  $R(A, B, C)$  und die Menge  $F = \{A \rightarrow B\}$ .

[Zur Lösung](#) [Zum Text](#)

## W | F 17

Gegeben sei folgende Menge von FD:  $F = \{A \rightarrow G, B \rightarrow EC, AB \rightarrow D\}$ . Ermitteln Sie mithilfe des Attributabschlusses, ob die FD  $B \rightarrow D$  aus  $F$  geschlussfolgert werden kann!

[Zur Lösung](#) [Zum Text](#)

## W | F 18

Geben Sie mithilfe des Attributabschlusses an, ob sich die funktionale Abhängigkeit  $C \rightarrow ABDE$  aus der Menge  $F = \{A \rightarrow C, C \rightarrow D, C \rightarrow B, B \rightarrow E\}$  ableiten lässt!

[Zur Lösung](#) [Zum Text](#)

## W | F 19

Gegeben seien die beiden Mengen von funktionalen Abhängigkeiten  $F = \{A \rightarrow BD, A \rightarrow C\}$  und  $G = \{A \rightarrow B, A \rightarrow D, D \rightarrow C\}$ . In welcher der drei Mengenbeziehungen ( $F^+ = G^+, F^+ \subset G^+, F^+ \supset G^+$ ) stehen die Abschlüsse von  $F$  und  $G$ ?

[Zur Lösung](#) [Zum Text](#)

## W | F 20

Gegeben sei das 1NF-Relationenschema  $R(A, B, C, D, E, F)$  mit den funktionalen Abhängigkeiten  $\{AB \rightarrow DE, D \rightarrow E, C \rightarrow F, AB \rightarrow A\}$ . Überführen Sie das Schema in die zweite Normalform!

[Zur Lösung](#) [Zum Text](#)

## W | F 21

Gegeben sind das Relationenschema  $R(A, B, C, D, E, F)$  in 1NF und die funktionalen Abhängigkeiten  $FD = \{C \rightarrow A, B \rightarrow E, E \rightarrow F\}$ . Überführen Sie dieses Schema zuerst in die 2NF und danach in die 3NF! Geben Sie dabei jeweils die verletzenen Abhängigkeiten der ursprünglichen Schemata und die Primärschlüssel und Abhängigkeiten der neuen Schemata an. Die Anzahl der 3NF-Relationen soll minimal sein!

[Zur Lösung](#) [Zum Text](#)

## W | F 22

Gegeben sei das Relationenschema  $R(A, B, C, D, E)$  in 1NF und die zugehörigen funktionalen Abhängigkeiten  $FD = \{AB \rightarrow C, D \rightarrow B, AB \rightarrow DE, AB \rightarrow B\}$ . Begründen Sie, ob die 2NF, 3NF oder BCNF für die Relation erfüllt oder nicht erfüllt sind!

[Zur Lösung](#) [Zum Text](#)

## W | F 23

Erweitern Sie die folgende Relation um Tupel, so dass die MVD  $B \twoheadrightarrow C$  erfüllt ist:

| $A$ | $B$ | $C$ | $D$ |
|-----|-----|-----|-----|
| 2   | 2   | 2   | 2   |
| 1   | 2   | 3   | 4   |

[Zur Lösung](#) [Zum Text](#)

## W | F 24

Legen Sie mittels einer SQL-Anweisung die Tabelle `WORKS_ON_DESCRIBED` an. Die Tabelle soll den gleichen Aufbau wie die Tabelle `WORKS_ON` haben (siehe [VL DB](#)), aber noch über eine zusätzliche Spalte `DESCRIPTION` verfügen. Denken Sie bei der Erzeugung der Tabelle auch an die Vergabe von Integritätsbedingungen!

[Zur Lösung](#) [Zum Text](#)

## W | F 25

Erzeugen Sie eine Tabelle *Student* mit den folgenden Attributen: 'Matrikelnummer', 'Name', 'Studiengang' und 'Anzahl benötigter Kreditpunkte'. Legen Sie geeignete Datentypen und Integritätsbedingungen für die Attribute fest.

[Zur Lösung](#) [Zum Text](#)

## W | F 26

Erzeugen Sie eine Tabelle *Student* mit den folgenden Attributen: 'Matrikelnummer', 'Name', 'Studiengang' und 'Anzahl benötigter Kreditpunkte'. Legen Sie geeignete Datentypen und Integritätsbedingungen für die Attribute fest.

[Zur Lösung](#) [Zum Text](#)

## W | F 27

Fügen Sie mithilfe von DML-Befehlen drei beliebige Studenten-Datensätze in die Tabelle *Student* (Aufgabe 26) ein. Verändern Sie anschließend für einen Eintrag den Studiengang mithilfe eines DML-Befehls. Der neue Studiengang des Studenten soll 'IMT' sein.

[Zur Lösung](#) [Zum Text](#)

## W | F 28

Fügen Sie mithilfe von DDL-Befehlen die zwei Spalten 'Geburtsdatum' und 'Lebensalter' zur Tabelle *Student* (Aufgabe 26) hinzu. Tragen Sie daraufhin mithilfe von DML-Befehlen entsprechende Geburtstage/Alter für alle Studenten in die Tabelle ein. Löschen Sie anschließend mithilfe eines DML-Befehls alle Studenten, die jünger als 18 Jahre sind.

[Zur Lösung](#) [Zum Text](#)

## W | F 29

Entfernen Sie mithilfe eines DDL-Befehls die in Aufgabe 4 hinzugefügte Spalte 'Alter' wieder. Löschen Sie nun alle Studenten in der Tabelle mithilfe eines DML-Befehls und entfernen Sie die Tabelle *Student* abschließend mithilfe eines DDL-Befehls.

[Zur Lösung](#) [Zum Text](#)

## W | F 30

Formulieren Sie eine SQL-Anfrage, welche die Nachnamen (`LNAME`) aller Angestellten, die am Projekt „Product X“ (`PNAME`) arbeiten, ausgibt (Tabellen: `EMPLOYEE`, `PROJECT`, `WORKS_ON` siehe [VL DB](#)).

[Zur Lösung](#) [Zum Text](#)



## W | F 31

Geben Sie für jeden Kurs den Titel (K\_TITEL), die Dauer (K\_DAUER), den Nettopreis (P\_TN1) in einer Spalte namens NETTO und den Bruttopreis in einer Spalte namens BRUTTO aus. Der Mehrwertsteuersatz beträgt dabei 19 % (Tabellen KURS, PREIS siehe [Uebung.sql](#)).

[Zur Lösung](#) [Zum Text](#)

## W | F 32

Die folgenden Anfragen ergeben eine Fehlermeldung, wenn Sie ausgeführt werden. Erläutern Sie den jeweiligen Fehler und korrigieren Sie die Anfragen entsprechend.

SQL

```
SELECT bdate
FROM employee, dependent
WHERE ssn = essn;
```

```
SELECT lname AS Angestellter, lname AS Vorgesetzter
FROM employee, employee
WHERE superssn = ssn;
```

[Zur Lösung](#) [Zum Text](#)

## W | F 33

Finden Sie alle Paare von Dozenten (Tabelle DOZENT siehe [Uebung.sql](#)), die den gleichen Vornamen (D\_VORNAME) haben. Geben Sie den Vornamen, die Dozentennummern (D\_NR) und beide Nachnamen (D\_NACHNAME) aus und ordnen Sie das Ergebnis nach Vornamen. Achten Sie darauf, dass die Dozenten nicht mit sich selbst verglichen werden. Achten Sie außerdem darauf, dass wenn ein Dozentenpaar (A, B) im Ergebnis auftritt, das Paar (B, A) nicht vorkommen soll (keine symmetrischen Paare).

[Zur Lösung](#) [Zum Text](#)

## W | F 34

Geben Sie die Nachnamen (LNAME) aller Angestellten aus, die Angehörige haben. Vermeiden Sie Duplikate bei der Ausgabe (Tabellen EMPLOYEE und DEPENDENT).

[Zur Lösung](#) [Zum Text](#)

## W | F 35

Nutzen Sie den ALL-Operator um die jüngsten sowie die ältesten Angestellten in Tabelle EMPLOYEE mit einer Anfrage zu bestimmen.

[Zur Lösung](#) [Zum Text](#)

## W | F 36

Nutzen Sie den ANY-Operator um alle Angestellten zu bestimmen, die eine Tochter haben (RELATIONSHIP). (Tabellen EMPLOYEE und DEPENDENT)

[Zur Lösung](#) [Zum Text](#)

## W | F 37

Finden Sie die Namen aller Angehörigen (DEPENDENT\_NAME) der Angestellten, die nicht am Projekt 'Computerization' arbeiten (Tabellen WORKS\_ON, PROJECT und DEPENDENT).

[Zur Lösung](#) [Zum Text](#)

## W | F 38

Erzeugen Sie die Tabelle `MEIN_PROJEKT`, die den gleichen Aufbau und Inhalt wie die Tabelle `PROJECT` hat. Anschließend soll mithilfe eines DML-Befehls eine Zeile verändert werden. Das Projekt mit dem Namen 'Reorganization' wurde an das Department 'Headquarters' abgegeben. Aktualisieren Sie den Eintrag des Projekts in Tabelle `MEIN_PROJEKT` und bestimmen Sie dazu die zugehörige Departmentnummer (`DNUMBER`, Tabelle `DEPARTMENT`) per Unterabfrage.

[Zur Lösung](#) [Zum Text](#)

## W | F 39

Gegeben sei die folgende Tabelle. Ermitteln Sie das Ergebnis der Anfrage `SELECT * FROM Table3 WHERE A >= B`; und erläutern Sie es!

**TABLE 3**

| A    | B    |
|------|------|
| 1    | NULL |
| NULL | 1    |
| 0    | 1    |
| 0    | 0    |
| NULL | NULL |
| 1    | 1    |

[Zur Lösung](#) [Zum Text](#)

## W | F 40

Ermitteln Sie alle Angestellten (`SSN` ausgeben), die keinen Vorgesetzten (`SUPERSSN`) haben (Tabelle `EMPLOYEE`).

[Zur Lösung](#) [Zum Text](#)

## W | F 41

Gegeben seien die folgenden Tabellen und Anfragen. Schreiben Sie das Ergebnis der Anfragen in Tabellenform auf und erläutern Sie die Unterschiede zwischen den verschiedenen JOIN-Arten.

**TABLE 1**

| A    | B | C |
|------|---|---|
| 1    | 3 | 6 |
| 2    | 4 | 7 |
| NULL | 5 | 8 |

**TABLE 2**

| C    | D | E |
|------|---|---|
| 1    | a | b |
| NULL | c | d |
| 2    | e | f |
| 2    | g | h |
| 8    | i | j |

- `SELECT * FROM Table1 t1, Table2 t2 WHERE t1.A = t2.C;`
- `SELECT * FROM Table1 t1 INNER JOIN Table2 t2 ON t1.A = t2.C;`
- `SELECT * FROM Table1 t1 FULL OUTER JOIN Table2 t2 ON t1.A = t2.C;`
- `SELECT * FROM Table1 t1 NATURAL JOIN Table2 t2;`

[Zur Lösung](#) [Zum Text](#)

## W | F 42

Gegeben sei die folgende SQL-Anfrage, welche die SSN aller Angestellten ermittelt, die keine Angehörigen haben. Formulieren Sie eine äquivalente SQL-Anfrage, die statt der MINUS-Operation einen entsprechenden Verbund/Join verwendet.

SQL

```
(SELECT ssn
FROM employee)
MINUS
(SELECT essn
FROM dependent);
```

[Zur Lösung](#) [Zum Text](#)

## W | F 43

Gegeben seien die folgende Tabelle und verschiedene Anfragen. Ermitteln Sie die Ergebnisse der Anfragen und erläutern Sie, wie diese zustande kommen.

TABLE 1

| x    |
|------|
| NULL |
| 0    |
| 1    |
| 2    |
| 1    |

TABLE 2

| a | b |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 1 | 2 |
| 3 | 2 |

- SELECT COUNT(\*) FROM Table1;
- SELECT COUNT(x) FROM Table1;
- SELECT SUM(x) FROM Table1;
- SELECT AVG(x) FROM Table1 WHERE x < 2;
- SELECT COUNT(b) FROM Table2 ORDER BY COUNT(b);
- SELECT a, COUNT(b) FROM Table2 GROUP BY a;
- SELECT a, COUNT(b) FROM Table2 GROUP BY a HAVING COUNT(b) < 2;

[Zur Lösung](#) [Zum Text](#)

## W | F 44

Geben Sie alle Kurse aus, die einen höheren Preis (P\_TN1) als den Durchschnitt aller Kurspreise haben! Nutzen Sie die Tabellen KURS und PREIS und verwenden Sie eine Unterabfrage zur Lösung der Aufgabe.

[Zur Lösung](#) [Zum Text](#)

## W | F 45

Bestimmen Sie das minimal und das maximal gezahlte Gehalt (SALARY) aller männlichen (SEX) Angestellten (Tabelle EMPLOYEE)!

[Zur Lösung](#) [Zum Text](#)

## W | F 46

Bestimmen Sie die Anzahl der Mitarbeiter und die Summe sowie den Durchschnitt ihrer jeweiligen Gehälter (**SALARY**) für jede Abteilung des Unternehmens! Geben Sie die Abteilungen jeweils mit ihrem Namen (**DNAME**) aus. (Tabellen **EMPLOYEE** und **DEPARTMENT**)

[Zur Lösung](#) [Zum Text](#)

## W | F 47

Geben Sie die Sozialversicherungsnummern (**SSN**) und Nachnamen (**LNAME**) aller Angestellten aus, die an mehr als 2 Projekten arbeiten (Tabellen **EMPLOYEE** und **WORKS\_ON**).

[Zur Lösung](#) [Zum Text](#)

## W | F 48

Ein Stadtgebiet ist eine Kombination aus Stadtname und Postleitzahl (**TN\_STADT**, **TN\_PLZ**). Geben Sie für jedes Stadtgebiet aus, wie viele Teilnehmer in diesem wohnen. Geben Sie dabei nur Stadtgebiete mit mehr als zehn Teilnehmern aus und sortieren Sie das Ergebnis absteigend nach der Anzahl der Teilnehmer. (Tabelle **TEILNEHMER** siehe [Uebung.sql](#))

[Zur Lösung](#) [Zum Text](#)

## W | F 49

Geben Sie die Anzahl Angehöriger für jeden einzelnen Angestellten der Firma aus! Nutzen Sie dazu einen geeigneten Verbund (Tabellen **EMPLOYEE** und **DEPENDENT**)! Die Anzahl der Angehörigen ergibt sich aus der Anzahl der Einträge in **DEPENDENT** für einen Angestellten. Hat ein Angestellter keine Einträge in **DEPENDENT**, so hat er 0 Angehörige.

[Zur Lösung](#) [Zum Text](#)

## W | F 50

Geben Sie für jede Abteilung ihre Abteilungsnummer (**DNUMBER**), ihren Namen (**DNAME**) und die Anzahl der Mitarbeiter aus! Im Ergebnis sollen nur Abteilungen erscheinen, in denen mindestens 3 Mitarbeiter mit verschiedenen Nachnamen arbeiten (Tabellen **EMPLOYEE** und **DEPARTMENT**).

[Zur Lösung](#) [Zum Text](#)

## W | F 51

Ermitteln Sie mithilfe eines SQL-Befehls alle Paare von Angestellten, die an den genau gleichen Projekten arbeiten. Geben Sie die Paare als Tupel der Form (**SSN**, **SSN**) aus. Vermeiden Sie Doppelungen: Ist (1,2) im Ergebnis, so soll (2,1) nicht im Ergebnis auftauchen. Beispiel: Person 1 arbeitet an Projekten X,Y und Z. Ergebnis der Anfrage sind dann alle Personen die ebenfalls an Projekten X,Y und Z (und keinen weiteren Projekten) arbeiten. (Tabellen **EMPLOYEE** und **WORKS\_ON**)

[Zur Lösung](#) [Zum Text](#)

## W | F 52

Gegeben seien folgende SQL-Anweisungen. Welche Zeilen werden durch die **UPDATE**-Anweisung verändert? Begründen Sie ihre Entscheidung!

SQL

```
CREATE VIEW personal AS
SELECT *
FROM employee
WHERE salary < 40000
WITH CHECK OPTION;

UPDATE personal
SET salary = 50000;
```

[Zur Lösung](#) [Zum Text](#)

W | F 53

Gegeben seien folgende SQL-Anweisungen. Lässt sich die Änderungsoperation problemlos ausführen? Begründen Sie Ihre Antwort!

SQL

```
CREATE VIEW projektpersonal AS
SELECT pname, fname, lname
FROM project, employee, works_on
WHERE pno = pnumber AND ssn = essn;

UPDATE projektpersonal
SET pname = 'Product A'
WHERE lname = 'Smith';
```

[Zur Lösung](#) [Zum Text](#)

W | F 54

Formulieren Sie die folgende SQL-Anfrage als Ausdruck der relationalen Algebra:

SQL

```
SELECT bv.BuchID, l.Nachname, l.Adresse
FROM Bibliothek b, BuchVerleih bv, Leiher l
WHERE b.Name = 'Datenbanken'
AND bv.Fristdatum = '10.02.2014'
AND b.BibID = bv.BibID
AND bv.KartenID = l.KartenID;
```

[Zur Lösung](#) [Zum Text](#)

W | F 55

Formulieren Sie die folgende SQL-Anfrage als Ausdruck der relationalen Algebra:

SQL

```
SELECT pno AS Projektnummer, SUM(salary) AS Projektkosten
FROM department, project
WHERE pno = pnumber
GROUP BY pno;
```

[Zur Lösung](#) [Zum Text](#)

W | F 56

Berechnen sie  $e(E) \leftarrow r(R) \div s(S)$  für die folgenden Relationen:

| $r(R)$ |       |     | $s(S)$ |
|--------|-------|-----|--------|
| A      | B     |     | A      |
| $a_1$  | $b_1$ | und |        |
| $a_2$  | $b_1$ |     |        |
| $a_3$  | $b_1$ |     |        |
| $a_4$  | $b_1$ |     |        |
| $a_1$  | $b_2$ |     | $a_2$  |
| $a_3$  | $b_2$ |     | $a_3$  |
| $a_2$  | $b_3$ |     |        |
| $a_3$  | $b_3$ |     |        |
| $a_4$  | $b_3$ |     |        |
| $a_1$  | $b_4$ |     |        |
| $a_2$  | $b_4$ |     |        |
| $a_3$  | $b_4$ |     |        |

[Zur Lösung](#) [Zum Text](#)

W | F 57

Berechnen sie  $e(E) \leftarrow r(R) \div \pi_A(r(R))$  für die folgende Relation:

| $r(R)$ |       |
|--------|-------|
| A      | B     |
| $a_1$  | $b_1$ |
| $a_2$  | $b_1$ |
| $a_3$  | $b_1$ |
| $a_4$  | $b_1$ |
| $a_1$  | $b_2$ |
| $a_3$  | $b_2$ |
| $a_2$  | $b_3$ |
| $a_3$  | $b_3$ |
| $a_4$  | $b_3$ |
| $a_1$  | $b_4$ |
| $a_2$  | $b_4$ |
| $a_3$  | $b_4$ |

[Zur Lösung](#) [Zum Text](#)

W | F 58

Gegeben sie alle Angestellten aus, die an allen Projekten arbeiten! Sowohl in SQL als auch in relationaler Algebra.

[Zur Lösung](#) [Zum Text](#)

W | F 59

Gibt der Ausdruck  $\{x \mid (\forall x)(\text{Person}(x) \wedge \text{CONDITION}(x))\}$  die Anfrage „Liefere alle Personen in einem beliebigen Datenbankschema zurück, die Bedingung CONDITION erfüllen.“ korrekt wieder?

[Zur Lösung](#) [Zum Text](#)

W | F 60

Finden Sie mithilfe einer Anfrage des relationalen Tupelkalküls den Namen (FNAME, LNAME) und das Gehalt (SALARY) von allen Angestellten, die in der Abteilung

'Research' (DNAME) arbeiten. (Tabellen EMPLOYEE und DEPARTMENT)

[Zur Lösung](#) [Zum Text](#)

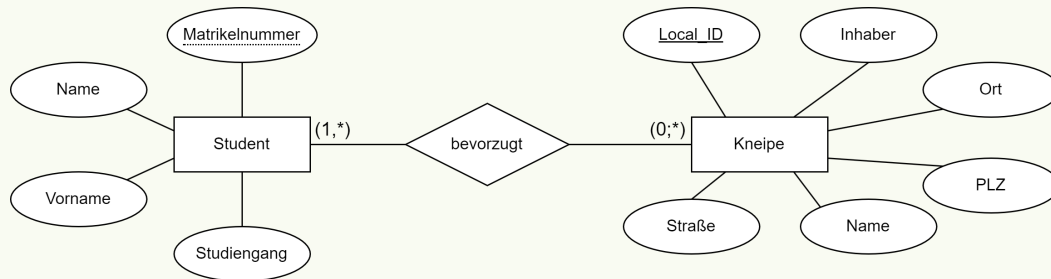
W | F 61

Finden Sie mithilfe einer Anfrage des relationalen Tupelkalküls die Namen aller Angestellten (FNAME, LNAME), die keine Angehörigen haben. (Tabellen EMPLOYEE und DEPENDENT)

[Zur Lösung](#) [Zum Text](#)

## 7.4 „Wahr oder Falsch“-Lösungen

### Lösung zu W | F 1

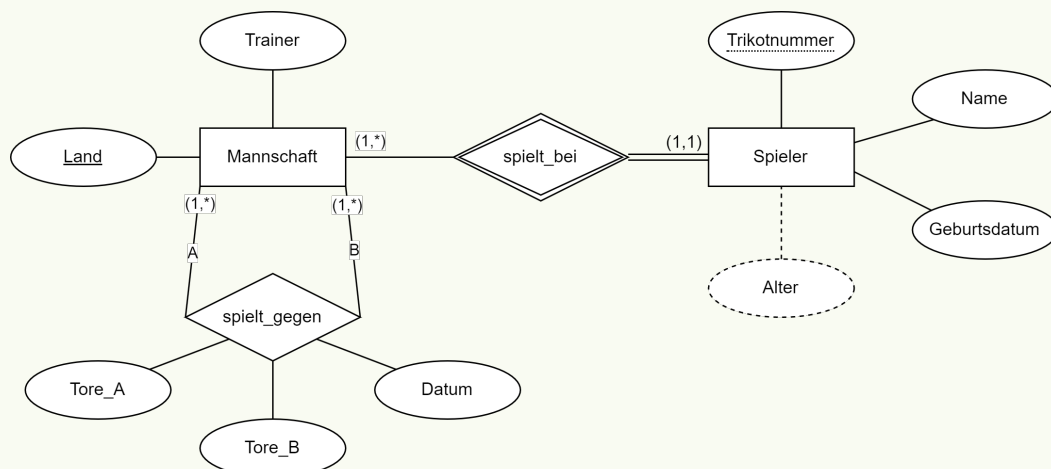


#### Bemerkung

- alle Studenten und alle Kneipen zu jeweils einer Entitätenmenge zusammenfassen
- Attribute aus dem Text identifizieren und den jeweiligen Entitätenmengen beifügen
- Primärschlüssel vergeben:
  - Matrikelnummer für Studenten (identifiziert jeden Studenten eindeutig)
  - Local\_ID für Kneipen  
(gleicher Inhaber könnte in der gleichen Straße zwei Kneipen mit gleichem Namen betreiben ...)
- binäre Beziehung zwischen Studenten und Kneipen in Beziehungsmenge „bevorzugt“ ausdrücken → Kardinalitäten festlegen (Anzahl beteiligter Entitäten beschränken):
  - ein Student hat **MINDESTENS EINE** (1,\*) bevorzugte Kneipe
  - jede Kneipe kann von **BELIEBIG VIELEN** (0,\*) bevorzugt werden

[Zur Frage](#) [Zum Text](#)

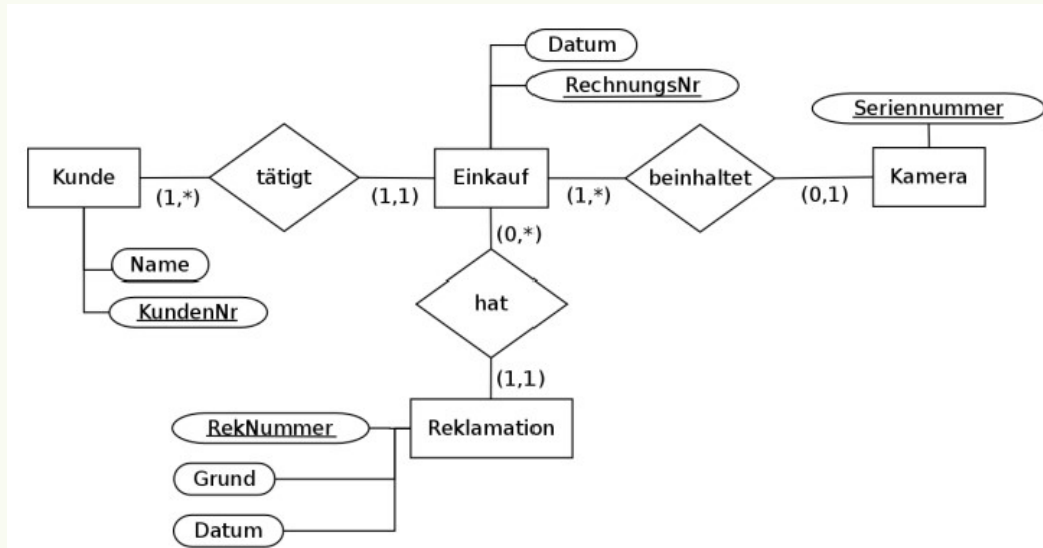
### Lösung zu W | F 2



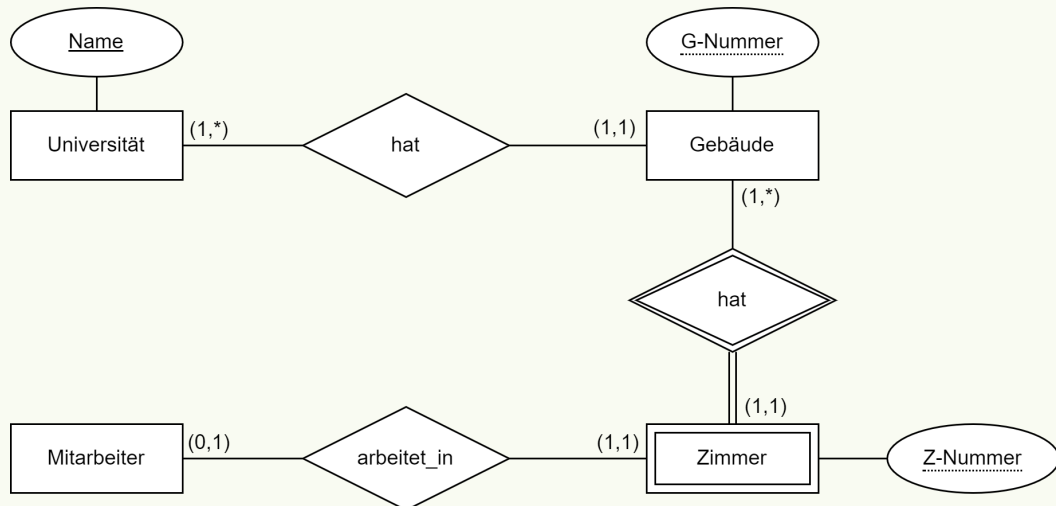
[Zur Frage](#) [Zum Text](#)



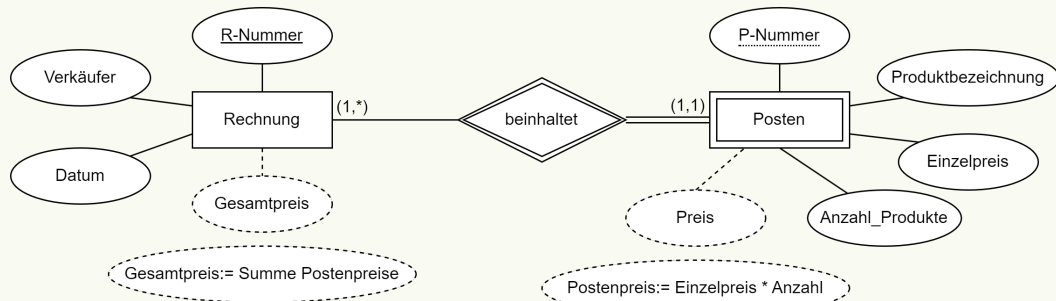
## Lösung zu W | F 3


[Zur Frage](#) [Zum Text](#)

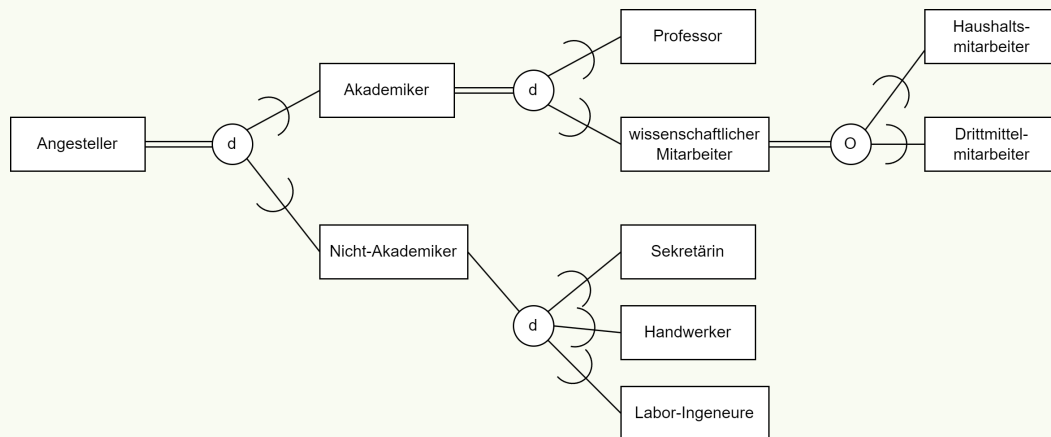
## Lösung zu W | F 4


[Zur Frage](#) [Zum Text](#)

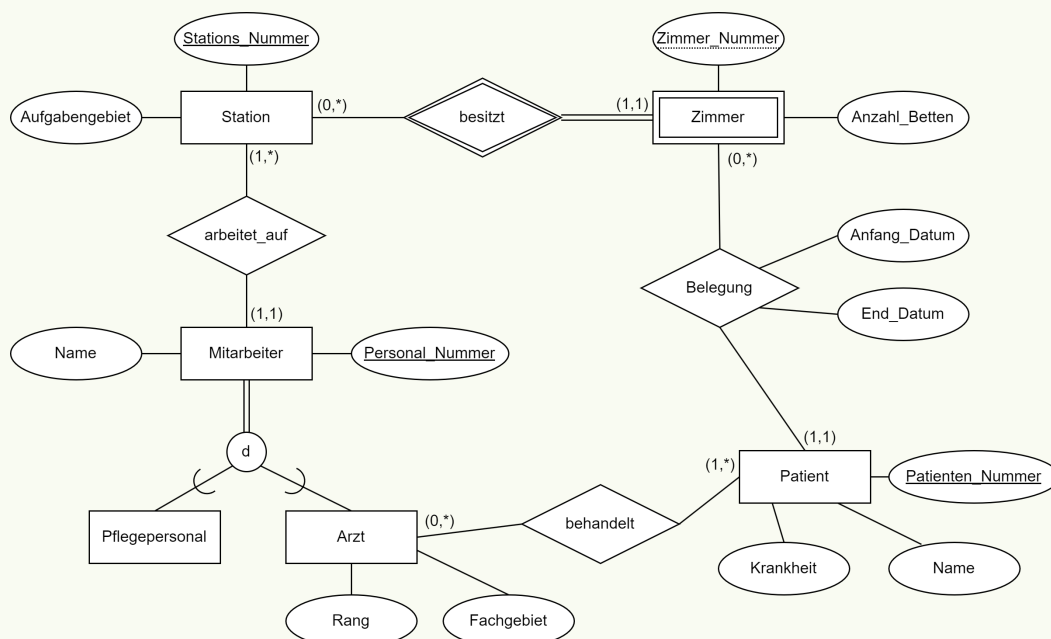
## Lösung zu W | F 5


[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 6


[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 7


[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 8

- Kneipe(Name, Inhaber, Straße, Ort, PLZ)
- Student(MatNr, Name, Vorname, Studiengang)
- mag(Name → Kneipe, MatNr → Student)

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 9

- Wohnheim(WNr, Adresse)
- Wohnheim-Hausmeister(WNr → Wohnheim, Hausmeister)
- Zimmer(ZNr, WNr → Wohnheim, Größe)
- Student(MatNr, Nachname, Vorname, Geburtsdatum, (ZNr, WNr) → Zimmer)

Zimmer ist schwach

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 10

- Kunde(KundenNr, Name)
- Einkauf(RechnungsNr, Datum, KundenNr → Kunde)
- Kamera(Seriennummer, RechnungsNr → Einkauf)
- Reklamation(RekNummer, Grund, Datum, RechnungsNr → Einkauf)

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 11

- Mannschaft(Name, Nummer, Gründungsjahr)
- spielt\_gegen(Name\_Heim → Mannschaft.Name, Nummer\_Heim → Mannschaft.Nummer, Name\_Gast → Mannschaft.Name, Nummer\_Gast → Mannschaft.Nummer, Tore\_Heim, Tore\_Gast)

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 12

- Partei(Name, Kürzel, #\_Mitglieder)
- Wahlbezirk(Bezirksnummer, Bezirksname)
- Wahlbezirk-Wahllokal(Bezirksnummer → Wahlbezirk, Wahllokal)
- Wähler(Wählernummer, Bezirksnummer → Wahlbezirk, Name → Partei)

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 13

## Option 1

- Mitarbeiter(Nummer, Name, Vorname)
- Akademisch(Nummer → Mitarbeiter, Grad)
- Nicht-Akademisch(Nummer → Mitarbeiter, Arbeitsbereich)

## Option 2

- Akademisch(Nummer, Name, Vorname, Grad)
- Nicht-Akademisch(Nummer, Name, Vorname, Arbeitsbereich)

## Option 3

Mitarbeiter(Nummer, Name, Vorname, Grad, Arbeitsbereich, Subklassen\_Name) mit Attribut Subklassen\_Name für die Zugehörigkeit

## Option 4

Mitarbeiter(Nummer, Name, Vorname, Grad, Arbeitsbereich, Ist\_Akademisch, Ist\_Nicht-Akademisch)

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 14

- Patient(PatientenNr, Name, Krankheit)
- Station(StationsNr, Aufgabengebiet)

- $\text{Zimmer}(\text{StationsNr} \rightarrow \text{Station}, \text{ZimmerNr}, \text{Bettenanzahl})$  (Zimmer ist schwach)
- $\text{Stationspersonal}(\text{PersonalNr}, \text{Name}, \text{StationsNr} \rightarrow \text{Station})$
- $\text{Arzt}(\text{PersonalNr} \rightarrow \text{Stationspersonal}, \text{Rang}, \text{Fachgebiet})$
- $\text{Pflegepersonal}(\text{PersonalNr} \rightarrow \text{Stationspersonal})$
- $\text{belegt}((\text{StationsNr}, \text{ZimmerNr}) \rightarrow \text{Zimmer}, \text{PatientenNr} \rightarrow \text{Patient}, \text{von}, \text{bis})$
- $\text{behandeln}(\text{PersonalNr} \rightarrow \text{Arzt}, \text{PatientenNr} \rightarrow \text{Patient})$

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 15

- $A \rightarrow B$ : ✓
- $B \rightarrow A$ : ✗ wegen  $3 \mapsto 2$  und  $3 \mapsto 3$
- $AB \rightarrow A$ : ✓ (Armstrong-Axiom 1, Reflexivität)
- $CD \rightarrow AB$ : ✗ wegen  $(4, 3) \mapsto (2, 3)$  und  $(4, 3) \mapsto (3, 3)$
- $ABCDE \rightarrow ABCDE$ : ✓ (trivial)
- $ABE \rightarrow ABD$ : ✗ wegen  $(2, 3, 1) \mapsto (2, 3, 2)$  und  $(2, 3, 1) \mapsto (2, 3, 3)$

→ die Relation verletzt die funktionale Abhängigkeit F

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 16

Aus  $R(A, B, C)$  und  $F = \{A \rightarrow B\}$  folgt:

$$F^+ = \left\{ \begin{array}{ll} A \rightarrow B & (\text{gegeben}) \\ \begin{array}{llllll} A \rightarrow A & B \rightarrow B & C \rightarrow C & AB \rightarrow AB & AC \rightarrow AC & BC \rightarrow BC \\ ABC \rightarrow ABC & AB \rightarrow A & AB \rightarrow B & AC \rightarrow A & AC \rightarrow C & BC \rightarrow B \\ BC \rightarrow C & ABC \rightarrow A & ABC \rightarrow B & ABC \rightarrow C & ABC \rightarrow AB & ABC \rightarrow AC \end{array} \\ ABC \rightarrow BC & (\text{Reflexivität}) \\ A \rightarrow AB & (\text{Vereinigung}) \\ \begin{array}{ll} AC \rightarrow ABC & AC \rightarrow BC \end{array} & (\text{Erweiterung}) \\ AC \rightarrow B & (\text{Zerlegung}) \end{array} \right\}$$

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 17

$$F = \{A \rightarrow G, B \rightarrow EC, AB \rightarrow D\}$$

$$B \rightarrow D \in F^+ \Leftrightarrow \{D\} \subseteq \{B\}^+ \text{ bzgl. } F$$

$$1. \{B\}^+ := \{B\}$$

$$2. B \rightarrow EC \in F \text{ und } \{B\} \subseteq \{B\}^+ \rightarrow \{B\}^+ := \{B, C, E\}$$

$$3. \{B\}^+ \text{ unverändert, also } \{B\}^+ \text{ endgültig}$$

$$B \rightarrow D \notin F^+, \text{ da } \{D\} \not\subseteq \{B\}^+ = \{B, C, E\}$$

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 18

$$F = \{A \rightarrow C, C \rightarrow D, C \rightarrow B, B \rightarrow E\}$$

$$C \rightarrow ABDE \in F^+ \Leftrightarrow \{A, B, D, E\} \subseteq \{C\}^+ \text{ bzgl. } F$$

1.  $\{C\}^+ := \{C\}$
  2.  $C \rightarrow D \in F$  und  $\{C\} \subseteq \{C\}^+ \rightarrow \{C\}^+ := \{C, D\}$
  3.  $C \rightarrow B \in F$  und  $\{C\} \subseteq \{C\}^+ \rightarrow \{C\}^+ := \{B, C, E\}$
  4.  $B \rightarrow E \in F$  und  $\{B\} \subseteq \{C\}^+ \rightarrow \{C\}^+ := \{B, C, D, E\}$
  5.  $\{C\}^+$  unverändert, also  $\{C\}^+$  endgültig
- $C \rightarrow ABDE \notin F^+$ , da  $\{A, B, D, E\} \not\subseteq \{C\}^+ = \{B, C, D, E\}$

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 19

- $F = \{A \rightarrow BD, A \rightarrow C\}$
- $\{A\}^+$  bzgl.  $F$ :  $\{A\}^+ = \{A, B, C, D\}$
  - $\{B\}^+$  bzgl.  $F$ :  $\{B\}^+ = \{B\}$
  - $\{C\}^+$  bzgl.  $F$ :  $\{C\}^+ = \{C\}$
  - $\{D\}^+$  bzgl.  $F$ :  $\{D\}^+ = \{D\}$
- $G = \{A \rightarrow B, A \rightarrow D, D \rightarrow C\}$
- $\{A\}^+$  bzgl.  $G$ :  $\{A\}^+ = \{A, B, D, C\}$
  - $\{B\}^+$  bzgl.  $G$ :  $\{B\}^+ = \{B\}$
  - $\{C\}^+$  bzgl.  $G$ :  $\{C\}^+ = \{C\}$
  - $\{D\}^+$  bzgl.  $G$ :  $\{D\}^+ = \{D, C\}$

$F^+ \subset G^+$ , da  $D \rightarrow C \notin F^+$

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 20

1NF-Relationenschema  $R(\underline{A}, \underline{B}, \underline{C}, D, E, F)$  mit  $F = \{AB \rightarrow DE, D \rightarrow E, C \rightarrow F, AB \rightarrow A\}$

Zerlegung in 2NF:  $R_1(\underline{A}, \underline{B}, \underline{C})$ ,  $R_2(\underline{A}, \underline{B}, D, E)$  und  $R_3(\underline{C}, F)$

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 21

1NF-Relationenschema  $R = (\underline{A}, \underline{B}, \underline{C}, \underline{D}, E, F)$  mit  $F = \{C \rightarrow A, B \rightarrow E, E \rightarrow F\}$

Zerlegung in 2NF:  $R_1 = (\underline{C}, \underline{D}, B, E, F)$  und  $R_2 = (\underline{C}, A)$

Zerlegung in 3NF:  $R_1 = (\underline{C}, \underline{D}, B)$ ,  $R_2 = (\underline{C}, A)$ ,  $R_3 = (\underline{B}, E)$  und  $R_4 = (\underline{E}, F)$

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 22

1NF-Relationenschema  $R = (\underline{A}, \underline{B}, C, D, E)$  mit  $F = \{AB \rightarrow C, D \rightarrow B, AB \rightarrow DE, AB \rightarrow B\}$

- 2NF: ✓ (keine funktionale Abhängigkeiten vom Teil des Schlüssels zu Nichtschlüsselattributen)
- 3NF: ✓ (keine transitiven Abhängigkeiten von Nichtschlüsselattributen zu einem Teil des Schlüssels)
- BCNF: ✗ ( $D \rightarrow B$  ist nicht trivial und  $D$  ist kein Superschlüssel)

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 23

|     |          |          |          |          |
|-----|----------|----------|----------|----------|
|     | <i>A</i> | <i>B</i> | <i>C</i> | <i>D</i> |
|     | 2        | 2        | 2        | 2        |
|     | 1        | 2        | 3        | 4        |
| neu | 1        | 2        | 2        | 4        |
|     | 2        | 2        | 3        | 2        |

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 24

SQL

```
CREATE TABLE WORKS_ON_DESCRIBED AS SELECT * FROM WORKS_ON;
ALTER TABLE WORKS_ON_DESCRIBED ADD DESCRIPTION VARCHAR(255);
ALTER TABLE WORKS_ON_DESCRIBED ADD PRIMARY KEY (ESSN, PNO);
ALTER TABLE WORKS_ON_DESCRIBED ADD FOREIGN KEY (ESSN)
REFERENCES EMPLOYEE(SSN);
ALTER TABLE WORKS_ON_DESCRIBED ADD FOREIGN KEY (PNO)
REFERENCES PROJECT(PNUMBER);
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 25

SQL

```
CREATE TABLE Student (
  MNr CHAR(7) NOT NULL,
  Name VARCHAR(30) NOT NULL,
  Studiengang VARCHAR(40),
  Anzahl_CP INT,
  PRIMARY KEY(MNr)
);
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 26

SQL

```
CREATE TABLE Student (
  MNr CHAR(7) NOT NULL,
  Name VARCHAR(30) NOT NULL,
  Studiengang VARCHAR(40),
  Anzahl_CP INT,
  PRIMARY KEY(MNr)
);
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 27

SQL

```
INSERT INTO Student VALUES ('5005801', 'Hai Tzung', 'KI', 180);
INSERT INTO Student VALUES ('5005000', 'Karl Toffel', 'Informatik', 180);
INSERT INTO Student VALUES ('5001000', 'Tom Tohmsen', 'BWL', 180);
```

```
UPDATE Student SET Studiengang='IMT' WHERE MNr='5001000';
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 28

```
SQL
ALTER TABLE Student ADD Geburtsdatum DATE;
ALTER TABLE Student ADD Lebensalter INT;

UPDATE Student
SET Geburtsdatum=DATE'2005-04-05',Lebensalter=19
WHERE MNr='5005801';

UPDATE Student
SET Geburtsdatum=DATE'2005-08-08',Lebensalter=19
WHERE MNr='5005000';

UPDATE Student
SET Geburtsdatum=DATE'2015-08-08',Lebensalter=9
WHERE MNr='5001000';

DELETE FROM Student
WHERE Lebensalter < 18;
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 29

```
SQL
ALTER TABLE Student DROP COLUMN Lebensalter;

DELETE FROM Student;

DROP TABLE Student;
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 30

```
SQL
SELECT LNAME
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE PNO=PNUMBER and ESSN=SSN and PNAME='Product X';
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 31

```
SQL
SELECT DISTINCT K_TITEL, K_DAUER,P_TN1 AS NETTO, P_TN1*1.19 AS BRUTTO
FROM KURS,PREIS
WHERE kurs.p_nr=preis.p_nr
ORDER BY K_TITEL ASC;
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 32

## Anfrage 1

- Fehler: Die Spalte bdate kommt sowohl in employee, als auch dependent vor. Die Spalte wurde somit nicht eindeutig abgefragt.
- Lösung: `SELECT employee.bdate...` oder `SELECT dependent.bdate...`

## Anfrage 2

- Fehler: Die Tabelle EMPLOYEE kommt 2x vor, wodurch bei Spalten nicht eindeutig ist, auf welche der beiden sich bezogen wird.
- Lösung: Umbenennung der beiden Tabellen:

SQL

```
SELECT E.lname AS Angestellter, S.lname AS Vorgesetzter
FROM employee E, employee S
WHERE E.superssn = S.ssn;
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 33

SQL

```
SELECT D1.d_vorname, D1.d_nr, D1.d_nachname, D2.d_nr, D2.d_nachname
FROM Dozent D1, Dozent D2
WHERE D1.D_NR < D2.D_NR and D1.d_vorname=D2.d_vorname
-- < filtert symmetrische Paare raus
ORDER BY D1.d_vorname;
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 34

SQL

```
SELECT DISTINCT employee.lname
FROM employee, dependent
WHERE ssn=essn;
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 35

SQL

```
SELECT lname, TRUNC(MONTHS_BETWEEN(SYSDATE, bdate) / 12) AS Age
FROM Employee
WHERE TRUNC(MONTHS_BETWEEN(SYSDATE, bdate)/12) <= ALL --jüngster
(SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, bdate) / 12)
FROM Employee)
OR
TRUNC(MONTHS_BETWEEN(SYSDATE, bdate)/12) >= ALL --ältester
(SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, bdate) / 12)
FROM Employee);
```

[Zur Frage](#) [Zum Text](#)



## Lösung zu W | F 36

SQL

```
SELECT LNAME, SSN
FROM EMPLOYEE
WHERE SSN = ANY (SELECT ESSN
                  FROM DEPENDENT
                  WHERE RELATIONSHIP='Daughter');
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 37

SQL

```
SELECT dependent_name
FROM dependent
WHERE not dependent.essn IN
(SELECT works_on.essn
 FROM works_on JOIN project ON works_on.pno=project.pnumber
 WHERE project.pname='Computerization');
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 38

SQL

```
-- Tabelle kopieren
CREATE TABLE Mein_Project AS SELECT * FROM project;
ALTER TABLE mein_project ADD PRIMARY KEY (pnumber);

UPDATE mein_project
SET dnum = (SELECT dnumber
            FROM department
            WHERE department.dname='Headquarters')
WHERE pname='Reorganization';
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 39

**TABLE 3**

| A    | B    |
|------|------|
| 1    | NULL |
| NULL | 1    |
| 0    | 1    |
| 0    | 0    |
| NULL | NULL |
| 1    | 1    |

→ SELECT \* FROM Table3 WHERE A &gt;= B; →

| A | B |
|---|---|
| 0 | 0 |
| 1 | 0 |

Erklärung:

Damit eine Zeile zum Ergebnis hinzugefügt wird, muss die WHERE-Bedingung auf TRUE auswerten, und sobald A oder B ein NULL-Wert haben, wird A >= B direkt UNKNOWN, was ungleich TRUE ist.

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 40

SQL

```
SELECT lname, ssn
FROM Employee
WHERE superssn IS NULL;
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 41

TABLE 1

| A    | B | C |
|------|---|---|
| 1    | 3 | 6 |
| 2    | 4 | 7 |
| NULL | 5 | 8 |

TABLE 2

| C    | D | E |
|------|---|---|
| 1    | a | b |
| NULL | c | d |
| 2    | e | f |
| 2    | g | h |
| 8    | i | j |

- `SELECT * FROM Table1 t1, Table2 t2 WHERE t1.A = t2.C;`

→

| t1.A | t1.B | t1.C | t2.D | t2.E |
|------|------|------|------|------|
| 1    | 3    | 6    | a    | b    |
| 2    | 4    | 7    | e    | f    |
| 2    | 4    | 7    | g    | h    |

→ normaler Verbund: Verbundbedingung in der **WHERE**-Klausel

- `SELECT * FROM Table1 t1 INNER JOIN Table2 t2 ON t1.A = t2.C;`

→

| t1.A | t1.B | t1.C | t2.D | t2.E |
|------|------|------|------|------|
| 1    | 3    | 6    | a    | b    |
| 2    | 4    | 7    | e    | f    |
| 2    | 4    | 7    | g    | h    |

→ (INNER) JOIN: Verbundbedingung in der **FROM**-Klausel

- `SELECT * FROM Table1 t1 FULL OUTER JOIN Table2 t2 ON t1.A = t2.C;`

→

| t1.A | t1.B | t1.C | t2.C | t2.D | t2.E |
|------|------|------|------|------|------|
| 1    | 3    | 6    | 1    | a    | b    |
| 2    | 4    | 7    | 2    | e    | f    |
| 2    | 4    | 7    | 2    | g    | h    |
| NULL | 5    | 8    | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | c    | d    |
| NULL | NULL | NULL | 8    | i    | j    |

→ FULL (OUTER) JOIN: Verbundbedingung in der **FROM**-Klausel und alle Tupel werden zurückgegeben (notfalls mit NULL aufgefüllt)

- `SELECT * FROM Table1 t1 NATURAL JOIN Table2 t2;`

→

| t1.A | t1.B | t1.C | t2.D | t2.E |
|------|------|------|------|------|
| NULL | 5    | 8    | i    | j    |

→ NATURAL JOIN: Verbundbedingung in der **FROM**-Klausel über Spalten mit gleichem Namen (notfalls Umbenennung)

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 42

SQL

```
SELECT ssn
FROM employee LEFT JOIN dependent ON employee.ssn = dependent.essn
WHERE dependent.essn IS NULL;
```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 43

TABLE 1

| x    |
|------|
| NULL |
| 0    |
| 1    |
| 2    |
| 1    |

TABLE 2

| a | b |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 1 | 2 |
| 3 | 2 |

- SELECT COUNT(\*) FROM Table1;

→

| COUNT(*) |
|----------|
| 5        |

→ COUNT(\*) zählt alle Zeilen (NULL inklusiv)

- SELECT COUNT(x) FROM Table1;

→

| COUNT(x) |
|----------|
| 4        |

→ COUNT(x) zählt alle Zeilen der Spalte x mit definierten Einträgen (NULL daher exklusiv)

- SELECT SUM(x) FROM Table1;

→

| SUM(x) |
|--------|
| 4      |

→ SUM(x) summiert alle Zeilen der Spalte x (NULL exklusiv)

- SELECT AVG(x) FROM Table1 WHERE x < 2;

→

| AVG(x)        |
|---------------|
| $\frac{2}{3}$ |

→ AVG(x) bestimmt den Mittelwert aller Zeilen der Spalte x, aber nur die Zeilen mit Einträgen kleiner 2, daher  $\frac{0+1+1}{3}$  (NULL exklusiv)

- SELECT COUNT(b) FROM Table2 ORDER BY COUNT(b);

→

| COUNT(b) |
|----------|
| 4        |

→ COUNT(b) zählt alle Zeilen der Spalte x mit definierten Einträgen (NULL daher exklusiv), die ORDER BY-Klausel hat keine Auswirkung, weil es nur einen Wert gibt.

- `SELECT a, COUNT(b) FROM Table2 GROUP BY a;`

→

| a | COUNT(a) |
|---|----------|
| 1 | 2        |
| 2 | 1        |
| 3 | 1        |

→ `GROUP BY a` fasst alle Zeilen mit selben Eintrag in Spalte a zu einer Äquivalenzklasse zusammen (Partitionierung), und für jede Gruppierung wird die Anzahl der Zeilen bestimmt (`COUNT(b)`)

- `SELECT a, COUNT(b) FROM Table2 GROUP BY a HAVING COUNT(b) < 2;`

→

| a | COUNT(a) |
|---|----------|
| 2 | 1        |
| 3 | 1        |

→ `GROUP BY a` fasst alle Zeilen mit selben Eintrag in Spalte a zu einer Äquivalenzklasse zusammen (Partitionierung), und für jede Gruppierung wird die Anzahl der Zeilen bestimmt (`COUNT(b)`), aber diese wird nur zum Ergebnis hinzugefügt, wenn die Anzahl kleiner 2 ist (`HAVING COUNT(b) < 2`)

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 44

SQL

```
SELECT *
FROM preis
WHERE preis.p_tn1 > (SELECT AVG(p_tn1)
                     FROM preis);
```

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 45

SQL

```
SELECT MAX(salary) AS Max_Gehalt, MIN(salary) AS Min_Gehalt
FROM employee
WHERE sex='M';
```

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 46

SQL

```
SELECT d.dname AS Department_Name, COUNT(*) AS Anzahl_Mitarbeiter,
SUM(e.salary) AS Gesamte_Ausgabe, AVG(e.salary) AS Durchschnittsgehalt
FROM employee e JOIN department d ON e.dno = d.dnumber
GROUP BY d.dname;
```

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 47

SQL

```
SELECT ssn, lname
FROM employee
WHERE ssn IN (SELECT essn
              FROM works_on
```

```
GROUP BY essn
HAVING COUNT(DISTINCT pno) > 2);
```

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 48

```
SQL
SELECT TN_STADT, TN_PLZ , COUNT(*) AS Anzahl_Teilnehmer
FROM TEILNEHMER
GROUP BY (TN_STADT, TN_PLZ)
HAVING COUNT(*) >10
ORDER BY (COUNT(*)) DESC;
```

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 49

```
SQL
SELECT E.SSN, E.LNAME, COUNT(D.ESSN)
FROM EMPLOYEE E LEFT JOIN DEPENDENT D ON E.SSN = D.ESSN
GROUP BY (E.SSN,E.LNAME);
```

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 50

```
SQL
SELECT D.DNUMBER,D.DNAME, COUNT(*)
FROM EMPLOYEE E JOIN DEPARTMENT D ON E.DNO = D.DNUMBER
GROUP BY (D.DNUMBER,D.DNAME)
HAVING COUNT(DISTINCT E.LNAME) >= 3;
```

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 51

```
SQL
SELECT DISTINCT ('(' || E1.SSN || ',' || E2.SSN || ')') AS SSN_Pair,
('(' || E1.LNAME || ',' || E2.LNAME || ')') AS Name_Pair,
W1.PNO AS ProjektNR
FROM (EMPLOYEE E1 JOIN WORKS_ON W1 ON E1.SSN = W1.ESSN) JOIN
      (EMPLOYEE E2 JOIN WORKS_ON W2 ON E2.SSN = W2.ESSN) ON W1.PNO = W2.PNO
WHERE E1.SSN < E2.SSN;
```

[Zur Frage](#) [Zum Text](#)

#### Lösung zu W | F 52

→ Fehlermeldung (und damit keine Änderung), weil die Bedingung der View **salary** < 40.000 ist, und mit dem WITH CHECK OPTION wird sichergestellt, dass nur Tupel eingefügt oder geändert werden können, die die Bedingung der View erfüllen. Daher ist das Update auf 50.000 nicht möglich.

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 53

→ Fehlermeldung, denn Views können nur dann aktualisiert werden, wenn sich die Änderung eindeutig auf eine zugrunde liegende Tabelle abbilden lässt. Wegen der Verbunde gibt es verschiedene Übersetzungen in Änderungen der zugrundeliegenden Tabellen:

- Sollen in Works\_On die Projekte, an denen Smith arbeitet, auf das Projekt 'Product A' geändert werden
- Soll der Name des Projekts, an dem Smith arbeitet, in der Projekt-Tabelle geändert werden

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 54

```

verbund ← (Bibliothek * BuchVerleih) * Leiher (2x Natural Join)
selektion_1 ←  $\sigma_{\text{BuchVerleih.Fristdatum}='10.02.2014'}(\text{verbund})$ 
selektion_2 ←  $\sigma_{\text{Bibliothek.Name}='Datenbanken'}(\text{selektion\_1})$ 
result ←  $\pi_{\text{BuchVerleih.BuchID, Leiher.Nachname, Leiher.Adresse}}(\text{selektion\_2})$ 

```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 55

```

verbund ← department  $\bowtie_{\text{pno} = \text{pnumber}}$  project (Theta-Join)
aggregiert(pno, SUM_salary) ←  $\text{pno} \mathcal{F}_{SUM} \text{ salary}(\text{verbund})$ 
result ←  $\rho_{\text{Projektnummer, Projektkosten}}(\text{aggregiert})$ 

```

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 56

| $r(R)$ |       |  |  |
|--------|-------|--|--|
| A      | B     |  |  |
| $a_1$  | $b_1$ |  |  |
| $a_2$  | $b_1$ |  |  |
| $a_3$  | $b_1$ |  |  |
| $a_4$  | $b_1$ |  |  |
| $a_1$  | $b_2$ |  |  |
| $a_3$  | $b_2$ |  |  |
| $a_2$  | $b_3$ |  |  |
| $a_3$  | $b_3$ |  |  |
| $a_4$  | $b_3$ |  |  |
| $a_1$  | $b_4$ |  |  |
| $a_2$  | $b_4$ |  |  |
| $a_3$  | $b_4$ |  |  |

 $\div$ 

| $s(S)$ |  |
|--------|--|
| A      |  |
| $a_2$  |  |
| $a_3$  |  |

 $=$ 

| $e(E)$ |  |
|--------|--|
| B      |  |
| $b_1$  |  |
| $b_3$  |  |
| $b_4$  |  |

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 57

| $r(R)$ |       |  |  |  |
|--------|-------|--|--|--|
| A      | B     |  |  |  |
| $a_1$  | $b_1$ |  |  |  |
| $a_2$  | $b_1$ |  |  |  |
| $a_3$  | $b_1$ |  |  |  |
| $a_4$  | $b_1$ |  |  |  |
| $a_1$  | $b_2$ |  |  |  |
| $a_3$  | $b_2$ |  |  |  |
| $a_2$  | $b_3$ |  |  |  |
| $a_3$  | $b_3$ |  |  |  |
| $a_4$  | $b_3$ |  |  |  |
| $a_1$  | $b_4$ |  |  |  |
| $a_2$  | $b_4$ |  |  |  |
| $a_3$  | $b_4$ |  |  |  |

 $\div$ 

| $\pi_A(r(R))$ |  |
|---------------|--|
| A             |  |
| $a_1$         |  |
| $a_2$         |  |
| $a_3$         |  |
| $a_4$         |  |

 $=$ 

| $e(E)$ |  |
|--------|--|
| B      |  |
| $b_1$  |  |

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 58

SQL:

SQL

```
SELECT ssn, lname
FROM employee
WHERE ssn IN (SELECT essn
              FROM works_on
              GROUP BY essn
              HAVING COUNT(DISTINCT pno) > (SELECT COUNT(pnumber)
                                           FROM Project));
```

relationale Algebra:

$$\begin{aligned} \text{all\_proj\_emps} &\leftarrow (\pi_{\text{essn}, \text{pno}}(\text{Works\_On})) \div (\pi_{\text{pnumber}}(\text{Project})) \\ \text{full\_emps} &\leftarrow \text{Employee} \bowtie_{\text{ssn} = \text{essn}} \text{all\_proj\_emps} \\ \text{result} &\leftarrow \pi_{\text{lname}}(\text{full\_emps}) \end{aligned}$$
[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 59

Der Ausdruck  $\{x \mid \forall x \text{ Person}(x) \wedge \text{CONDITION}(x)\}$  ist wegen des Allquantors **falsch**: Der Allquantor bezieht sich auf alle Elemente der Domäne. Damit ein Tupel also in der Anfrage enthalten ist, müsste für alle Tupel  $x$  in der Datenbank gelten, dass sie eine Person sind und die Bedingung erfüllen.

Falls es in der Datenbank jedoch Tupel gibt, die keine Personen sind, dann ist die Bedingung falsch, weil es dann mindestens ein  $x$  gibt, für das  $\text{Person}(x)$  nicht gilt. Das bedeutet im Umkehrschluss, dass die Anfrage unabhängig von der anderen Bedingung immer leer sein wird, weil die Bedingung  $\text{Person}(x)$  nie global für alle  $x$  erfüllt sein kann.

[Zur Frage](#) [Zum Text](#)

## Lösung zu W | F 60

$$\{e.FNAME, e.LNAME, e.SALARY \mid \exists e \in \text{Employee} \wedge \exists d \in \text{Department} \wedge e.DNO = d.DNUMBER \wedge d.DNAME = \text{'Research'}\}$$

Oder

$$\{e.FNAME, e.LNAME, e.SALARY \mid Employee(e) \wedge (\exists d)(Department(d) \wedge e.DNO = d.DNUMBER \wedge d.DNAME = 'Research')\}$$

[Zur Frage](#) [Zum Text](#)

Lösung zu W | F 61

$$\{e.FNAME, e.LNAME \mid \exists Employee(e) \wedge (\neg(\exists d)(Dependent(d) \wedge d.ESSN = e.SSN))\}$$

Oder

$$\{e.FNAME, e.LNAME \mid Employee(e) \wedge ((\forall d)(\neg(Dependent(d)) \vee \neg(e.SSN = d.ESSN)))\}$$

[Zur Frage](#) [Zum Text](#)