Grundlagen: Datenbanken

1. Zentralübung

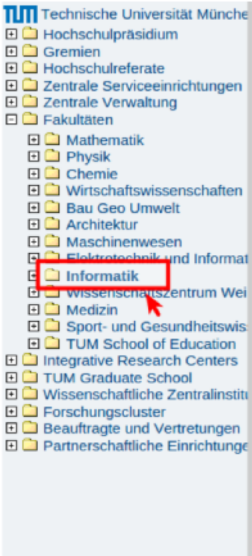
Harald Lang

Changelog:

- Folie "Einschub: Relationale Division": fehlende Projektion ergänzt
- Folie "Höchste NF bestimmen" Aufgabe d): R ist in 4.NF, da AB Superschlüssel
- Folie "Mehrwertige Abhängigkeiten": Nummerierung der MVDs vertauscht

FAQs

- **Ist der Prüfungtermin schon bekannt?**
  - Termin: **Mi. 18.02.2015, 08:00 Uhr**



FAQs

- **Gilt der Bonus auch für die Nachholklausur?**
  - Ja. Selbst dann, wenn die Hauptklausur nicht bestanden wurde.
- **Ich kann nicht zur ersten Klausur antreten. Muss ich zur Hauptklausur angemeldet sein, um an der Nachholklausur teilnehmen zu dürfen?**
  - Bitte nicht, das spart Papier und schont die Umwelt.
- **Ist der Termin für die Nachholklausur schon bekannt?**
  - Noch nicht. Voraussichtlich wird der Termin Ende März/Anfang April sein.

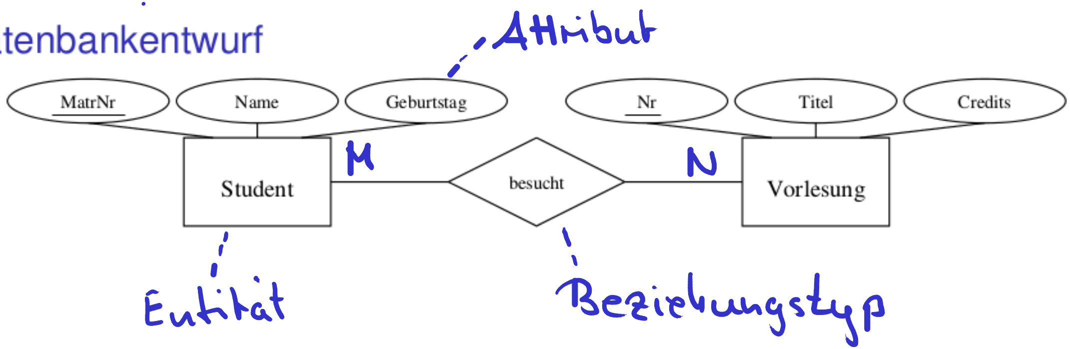
Diese Folien werden online gestellt.

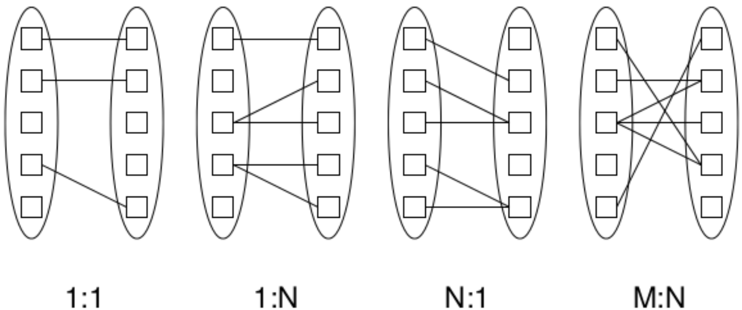
Nutzen Sie die heutige Veranstaltung um Fragen zu stellen. :)

Agenda

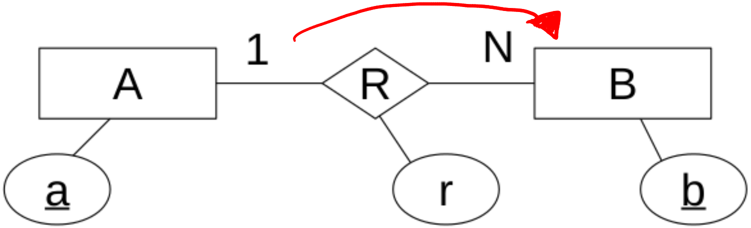
- ER-Modellierung
- Relationales Modell
- Anfragesprachen (Algebra & Kalküle)
- SQL
- Relationale Entwurfstheorie
- (Physische Datenorganisation)

Datenbankentwurf





ER-Modell in Schema überführen und verfeinern



$A : \{[\underline{a}]\}$

$B : \{[\underline{b}]\}$

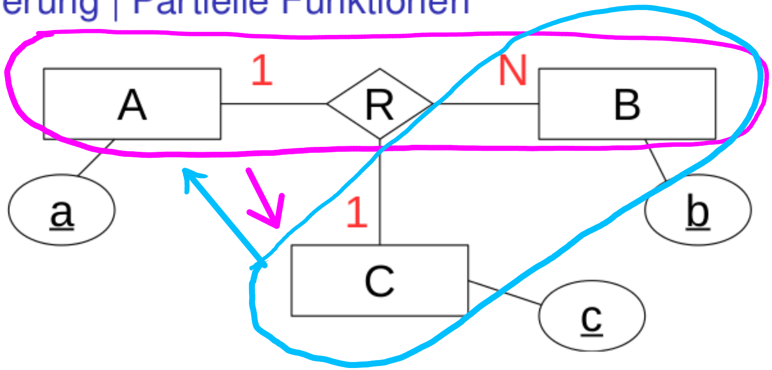
$R : \{[a, \underline{b}, r]\}$

<u>Verfeinern</u>:

$A : \{[\underline{a}]\}$

$B : \{[\underline{b}, a, r]\}$

ER-Modellierung | Partielle Funktionen



Partielle Funktionen:

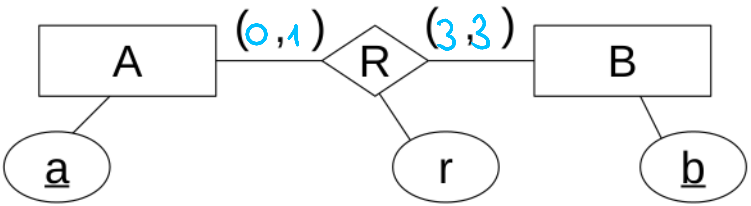
$A \times B \rightarrow C$

$B \times C \rightarrow A$

Relationen:

$R: \{[\underline{a}, \underline{b}, \underline{c}]\}$

(Min,Max) - Angaben



	R			
	a	b	r	
1x	1	b1	...	3x
	2	b1	...	
	3	b1		
	4	b2		3x
	:	b2		
	:	b2		

Das Relationale Modell

Definition

- Eine relationale Datenbank enthält eine Menge von Relationen
- Eine Relation R besteht aus zwei Bestandteilen:
  - Einer **Instanz** R: eine Tabelle mit Zeilen und Spalten; der *aktuelle Inhalt* der Relation (auch Ausprägung genannt)
  - Einem **Schema** $\mathcal{R}$: spezifiziert den *Namen der Relation* und die *Namen und Datentypen der Spalten*; legt die Struktur der Relation fest

Das Relationale Modell

Beispielausprägung:

Name der Relation

Attribute

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	10
27550	Schopenhauer	6
...	...	...

Tupel

Schema:

Studenten : {[MatrNr : integer, Name : string, sem : integer]}

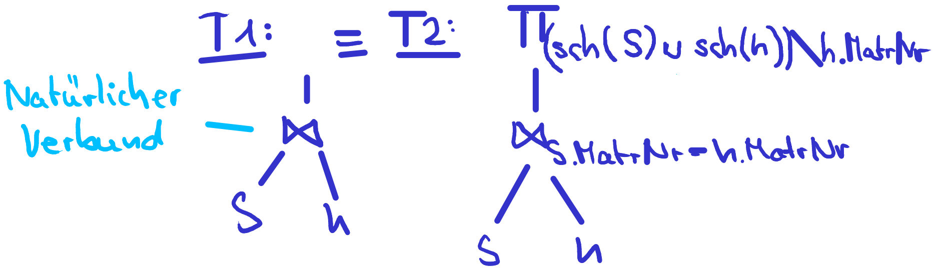
Studenten: {[MatrNr, Name, sem]}

oft werden die Typen nicht benötigt

Relationale Algebra

Algebraische Operatoren:

Projektion	$\Pi_{A_1,\ldots,A_n}$
Selektion	σ_p
Kreuzprodukt	$\times$
Verbund (Join)	$\bowtie_\theta$, ⟕$_\theta$, ⟖$_\theta$, ⟗$_\theta$, $\ltimes_\theta$, $\rtimes_\theta$, $\rhd_\theta$, $\lhd_\theta$
Mengenoperationen	$\cup, \cap, \setminus$
Division	$\div$
(Gruppierung/Aggregation)	$\Gamma_{A_1,\ldots,A_n\,;\,a_1:f_1,\ldots,a_m:f_m}$
Umbenennung	ρ_N, oder $\rho_{a_1 \leftarrow b_1,\ldots,a_n \leftarrow b_n}$



Relationale Algebra

Finde Studenten (nur Namen ausgeben), die im gleichen Semester sind wie Feuerbach.

$\pi_{s2.Name}$

$\bowtie_{s1.sem=s2.sem \wedge s1.Name \neq s2.Name}$

$\sigma_{s1.Name='Feuerbach'}$

ρ_{s1}

Studenten

ρ_{s2}

Studenten

S1

MatrNr	Name	Sem
4711	FB	2

Tupelkalkül

Finde Studenten, die die Vorlesung Grundzüge hören.

„für das gilt“

binden

Projektion

$$\{ s \mid s \in \text{Studenten} \wedge \exists h \in \text{hören} (s.\text{MatrNr} = h.\text{MatrNr} \wedge \exists v \in \text{Vorlesungen} (v.\text{VorlNr} = h.\text{VorlNr} \wedge v.\text{Titel} = \text{'Grundzüge'}))\}$$

Gültigkeits-
bereich von
h

nur den Namen projizieren : $\{ [s.\text{Name}] \mid \ldots \}$

Tupelkonstruktor

Domänenkalkül

Finde Studenten, die die Vorlesung Grundzüge hören.

Bindung: $\{[\underbrace{\text{MatrNr}}_{m}, \underbrace{\text{Name}}_{n}, \underbrace{\text{Semester}}_{s}]\}$

$$\{[m,n,s] \mid [m,n,s] \in \text{Studenten} \wedge \exists v([m,v] \in \text{hören} \wedge \exists t, sws, g([v,t,sws,g] \in \text{Vorlesungen} \wedge t = \text{'Grundzüge'}))\}$$

! Alle Domänenvariablen tauchen entweder in der Projektion auf oder wurden explizit quantifiziert. !

ALL-Quantifizierung

Finde Studenten, die ALLE Vorlesungen gehört haben.

Tupelkalkül:

$\{ s \mid s \in S^{\text{tudenten}} \wedge \underbrace{\forall v \in V^{\text{orlesungen}}}_{\text{die Tupelvariable kann an nur an Tupel gebunden werden, die auch in V vorkommen}} (\exists h \in \text{hören} (h.\text{MatrNr} = s.\text{MatrNr} \wedge v.\text{VorlNr} = h.\text{VorlNr}) \}$

ALL-Quantifizierung

Finde Studenten, die ALLE Vorlesungen gehört haben.

Domänenkalkül:

WT:

=>	w	f
w	w	f
f	w	w

beliebige Werte (ungebunden)

„nicht alle sind in V enthalten"

Implikation

! $\{[m,n,s] \mid [m,n,s] \in S \wedge \forall v,t,sws,g([v,t,sws,g] \in V \Rightarrow [m,v] \in \text{hören})\}$

nur dann, wenn v,t,sws,g ein Tupel „formen", das in V enthalten ist

kann zu „falsch" evaluieren, wodurch der Ausdruck wahr wird.

dann muss auch ein entsprechendes Tupel in hören vorhanden sein

ALL-Quantifizierung

Finde Studenten, die ALLE Vorlesungen gehört haben.

Relationale Algebra:

relationale Division

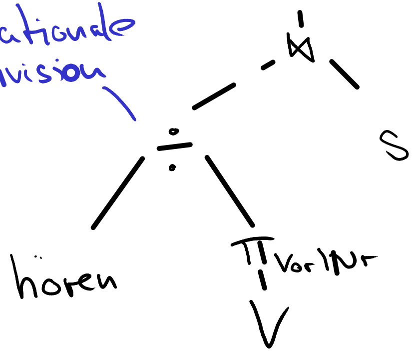
MatrNr	VorlNr
1	222
1	333
1	444
2	222
2	444

÷

VorlNr
222
333
444

=

MatrNr
1



Einschub: Relationale Division

$R \div S = \Pi_{\mathcal{R}\setminus\mathcal{S}}(R) \setminus \Pi_{\mathcal{R}\setminus\mathcal{S}}((\Pi_{\mathcal{R}\setminus\mathcal{S}}(R) \times S) \setminus R)$

! Voraussetzung: $\mathcal{R} \supseteq \mathcal{S}$!

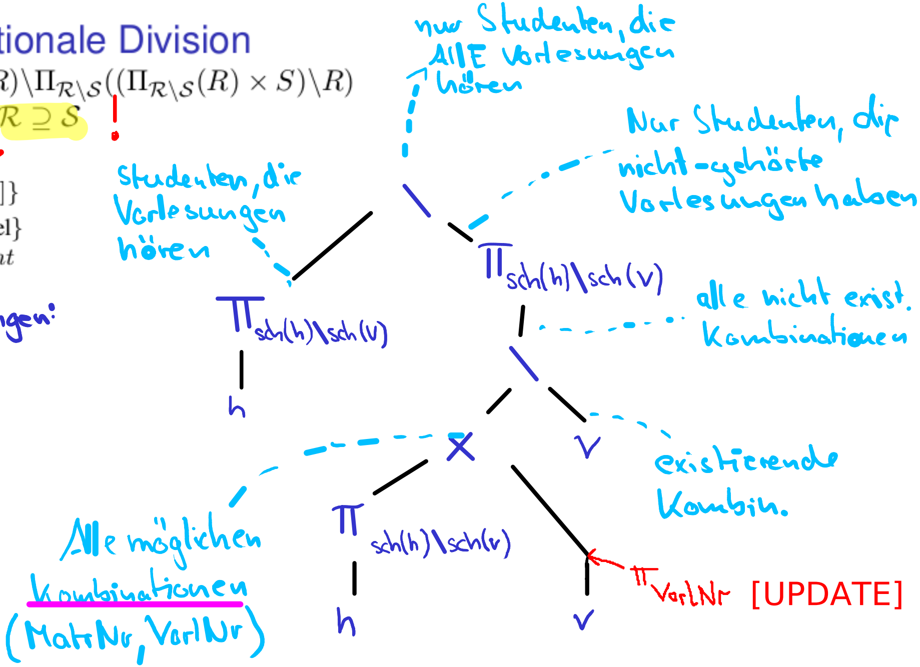
Beispiel:

$Studenten$: {[Name]}

$besucht$: {Name,Titel}

$Studenten \div besucht$

hören ÷ Vorlesungen:



ALL-Quantifizierung

SQL:

allquantifizierte Ausdrücke umformen
in existenzquantifizierende Ausdrücke

Finde Studenten, die ALLE Vorlesungen gehört haben.

$\forall x (p) \equiv \neg \exists x (\neg p)$

| es existiert kein x, für das p nicht gilt

Prädikat, muss für alle x gelten

SQL: kein ALL, lediglich EXISTS (und NOT EXISTS)

$\{s \mid s \in S \wedge \forall v \in V (\exists h \in hören (h.MatrNr = s.MatrNr \wedge v.VorlNr = h.VorlNr))\}$

$\Leftrightarrow \{s \mid s \in S \wedge \neg(\exists v \in V (\neg(\exists h \in hören (h.MatrNr = s.MatrNr \wedge v.VorlNr = h.VorlNr))))\}$

$\neg(A \Rightarrow B)$

$\equiv \neg A \vee B$

Tupelvariable

```
select s.*
from Studenten s
where not exists ( select *
                   from Vorlesungen v
                   where not exists ( select *
                                      from hoeren h
                                      where h.MatrNr = s.MatrNr
                                        and v.VorlNr = h.VorlNr ))
```

Vorlesung, die nicht gehört wird

Student für den es keine Vorlesung gibt, die er nicht gehört hat

Andere Join Arten | Semi-Join

$$Q_1 : Vorlesungen \ltimes hoeren$$

SQL (mit EXISTS):

```
select *
from vorlesungen v
where exists (
  select * from hoeren h
  where h.vorlnr = v.vorlnr)
```

SQL (ohne EXISTS):

```
select distinct v.*
from vorlesungen v, hoeren h
where h.vorlnr = v.vorlnr
```

Andere Join Arten | Outer-Join

$$Q : R ⟖ S, \qquad \text{mit } R = \{a, b, c\} \text{ und } S = \{c, d, e\}$$

$R \bowtie S$

$=$

Domänenkalkül:

$$Q_1 : \{[a, b, c, d, e] \mid [a, b, c] \in R \wedge [c, d, e] \in S\}$$

$$Q_2 : \{[\underbrace{a : null, b : null}_{\text{NULL-Werte}}, c, d, e] \mid [c, d, e] \in S \wedge \nexists a', b' ([a', b', c] \in R)\}$$

alle Tupel aus S, die keinen Join-Partner in R haben

$$Q = Q_1 \cup Q_2$$

Relationale Entwurftheorie

Funktionale Abhängigkeiten (kurz FDs, für functional dependencies):

- Seien α und β Attributmengen eines Schemas $\mathcal{R}$.
- Wenn auf $\mathcal{R}$ die FD $\alpha \rightarrow \beta$ definiert ist, dann sind nur solche Ausprägungen R zulässig, für die folgendes gilt:
  - Für alle Paare von Tupeln $r, t \in R$ mit $r.\alpha = t.\alpha$ muss auch gelten $r.\beta = t.\beta$.

MatrNr → Namen

Funktionale Abhängigkeiten

trivial

Seien $\alpha,\beta,\gamma,\delta \in \mathcal{R}$

Axiome von Armstrong:

- *Reflexivität:*
  Falls $\beta \subseteq \alpha$, dann gilt immer $\alpha \rightarrow \beta$
- *Verstärkung*:
  Falls $\alpha \rightarrow \beta$ gilt, dann gilt auch $\alpha\gamma \rightarrow \beta\gamma$
- *Transitivität*:
  Falls $\alpha \rightarrow \beta$ und $\beta \rightarrow \gamma$ gelten, dann gilt auch $\alpha \rightarrow \gamma$

Mithilfe dieser Axiome können alle *geltenden* FDs hergeleitet werden.

Sei F eine FD-Menge. Dann ist F^+ die Menge aller geltenden FDs (Hülle von F)

Funktionale Abhängigkeiten

Nützliche und vereinfachende Regeln:

- *Vereinigungsregel*:
  Falls $\alpha \to \beta$ und $\alpha \to \gamma$ gelten, dann gilt auch $\alpha \to \beta\gamma$
- *Dekompositionsregel*:
  Falls $\alpha \to \beta\gamma$ gilt, dann gilt auch $\alpha \to \beta$ und $\alpha \to \gamma$
- *Pseudotransitivitätsregel*:
  Falls $\alpha \to \beta$ und $\gamma\beta \to \delta$ gelten, dann gilt auch $\gamma\alpha \to \delta$

Schlüssel

- Schlüssel identifizieren jedes Tupel einer Relation $\mathcal{R}$ eindeutig.
- Eine Attributmenge $\alpha \subseteq \mathcal{R}$ ist ein **Superschlüssel**, gdw. $\alpha \to \mathcal{R}$
- Ist α zudem noch *minimal*, ist es auch ein **Kandidatenschlüssel** (meist mit κ bezeichnet)
  - Es existiert also kein $\alpha' \subset \alpha$ für das gilt: $\alpha' \to \mathcal{R}$

- I.A. existieren mehrere Super- und Kandidatenschlüssel.
- Man muss sich bei der Realisierung für einen Kandidatenschlüssel entscheiden, dieser wird dann **Primärschlüssel** genannt.
- Der triviale Schlüssel $\alpha = \mathcal{R}$ existiert immer.

$\mathcal{R} \to \mathcal{R}$

Schlüssel bestimmen

- Ob ein gegebenes α ein Schlüssel ist, kann mithilfe der Armstrong Axiome ermittelt werden (zu Aufwendig!)
- Besser: Die **Attributhülle** $AH(\alpha)$ bestimmen.
- Beispiel: $\mathcal{R} = \{ A , B , C , D \}$,

  mit den FDs $F_{\mathcal{R}} = \{AB \to CD, B \to C, D \to B\}$

  $AH(\{\underline{D}\})$: $D \overset{3}{\rightsquigarrow} B,D \overset{2}{\rightsquigarrow} BCD$

  $AH(\{A, D\})$: $AD \overset{3}{\rightsquigarrow} ABD \overset{1}{\rightsquigarrow} ABCD = \mathcal{R}$ => AD ist Superschlüssel

Mehrwertige Abhängigkeiten

multivalued dependencies (MVDs)

"Halb-formal":

- Seien α und β disjunkte Teilmengen von $\mathcal{R}$
- und $\gamma = (\mathcal{R} \backslash \alpha) \backslash \beta$
- dann ist β mehrwertig abhängig von α $(\alpha \twoheadrightarrow \beta)$, wenn in jeder gültigen Ausprägung von $\mathcal{R}$ gilt:
- Bei zwei Tupeln mit gleichem α-Wert kann man die β-Werte vertauschen, und die resultierenden Tupel müssen auch in der Relation enthalten sein.

Wichtige Eigenschaften:

- Jede FD ist auch eine MVD (gilt i.A. nicht umgekehrt)
- wenn $\alpha \twoheadrightarrow \beta$, dann gilt auch $\alpha \twoheadrightarrow \gamma$ (Komplementregel)
- $\alpha \twoheadrightarrow \beta$ ist trivial, wenn $\beta \subseteq \alpha$ ODER $\alpha \cup \beta = \mathcal{R}$ (also $\gamma = \emptyset$)

Mehrwertige Abhängigkeiten

Beispiel: $R = \{\text{Professor}, \text{Vorlesung}, \text{Assistent}\}$

Professor	Vorlesung	Assistent
TN	GDB	Viktor
TN	QO	Viktor
TN	GDB	Harald
TN	QO	Harald
TN	DB impl	Viktor
TN	DBimpl	Harald

MVD (1) $\underbrace{\text{Prof}}_{\alpha} \twoheadrightarrow \underbrace{\text{Vorl}}_{\beta}$

$(R \setminus \alpha) \setminus \beta = \gamma = \{\text{Assi}\}$

MVD 1 und 2 sind komplementär

+ Assi 'Harald' → 2 neue Tupel

+ Vorl 'DBimpl'

Normalformen: 1NF $\supset$ 2NF $\supset$ 3NF $\supset$ BCNF $\supset$ 4NF

$R: \{[A, \{B_1, B_2\}, C\}$ ↯

- **1. NF**: Attribute haben nur atomare Werte, sind also nicht mengenwertig.
- **2. NF**: Jedes Nichtschlüsselattribut (NSA) ist voll funktional abhängig von jedem Kandidatenschlüssel.
  - β hängt **voll funktional** von α ab ($\alpha \xrightarrow{\bullet} \beta$), gdw. $\alpha \to \beta$ und es existiert kein $\alpha' \subset \alpha$, so dass $\alpha' \to \beta$ gilt.
- **3. NF**: Frei von transitiven Abhängigkeiten. Alle NSAe hängen direkt von einem Kandidatenschlüssel ab.

  $A \to B \to C$ ↯

  - für alle geltenden nicht-trivialen FDs $\alpha \to \beta$ gilt entweder
    - α ist ein Superschlüssel, oder
    - jedes Attribut in β ist in einem Kandidatenschlüssel enthalten
- **BCNF**: Die linken Seiten (α) aller geltenden nicht-trivalen FDs sind Superschlüssel.
- **4. NF**: Die linken Seiten (α) aller geltenden nicht-trivalen MVDs sind Superschlüssel.

Höchste NF bestimmen

a) $\mathcal{R} = \{A, B, C\}, F_{\mathcal{R}} = \{AB \to C, B \to C\}$

Kandidatenschlüssel

Nichtschlüsselattribute

1NF ✓

2NF ✗ $\mathcal{K} = \{A, B\}$ NSA: $\mathcal{R} \setminus \mathcal{K} = \{C\}$ $\mathcal{K} \dot{\to} C$?

nein

$AB \to C$

$B \to C$

b) $\mathcal{R} = \{A, B, C\}, F_{\mathcal{R}} = \{A \to B, B \to C\}$

1NF ✓

2NF ✓ $\mathcal{K} = \{A\} \dot{\to} \mathcal{R}$

3NF ✗ $A \to \underbrace{B \to C}_{\text{trans.}}$

bzw

die geltende FD

$\underbrace{B}_{\alpha} \to \underbrace{C}_{\beta}$ verletzt die 3.NF

kein Superschlüssel

nicht in einem Kandidatenschl. enthalten

Höchste NF bestimmen

c) $\mathcal{R} = \{A, B, C, D\}$, $F_{\mathcal{R}} = \{\underline{A}B \rightarrow D,\ B \rightarrow C,\ C \rightarrow B\}$

1NF ✓

2NF ✓ $\mathcal{K}_1 = \{A B\}$ $\mathcal{K}_2 = \{A, C\}$ NSA: $\{D\}$ $\forall \mathcal{K}_i \, (\mathcal{K}_i \xrightarrow{\bullet} \{D\})$ ✓

3NF ✓ NSA $\{D\}$ wird direkt bestimmt durch $\mathcal{K}_1$

BCNF ✗ $B \rightarrow C$ verletzt BCNF. B ist kein Superschlüssel. (analog $C \rightarrow B$)

d) $\mathcal{R} = \{A, B, C, D\}$, $F_{\mathcal{R}} = \{AB \rightarrow CD\}$

1NF ✓

2NF ✓ $\mathcal{K} = \{A, B\} \xrightarrow{\bullet} CD$ NSA

3NF ✓ kein transitiven Abhängigkeiten

BCNF ✓ die einzigen geltenden nicht-trivialen FDs sind

$\boxed{AB} \rightarrow CD$

$\boxed{AB} \rightarrow C$

$\boxed{AB} \rightarrow D$

Superschlüssel (sogar Kand.schl.)

4NF ✓ die aus $AB \rightarrow C$ abgeleitete MVD $AB \twoheadrightarrow C$ verletzt die 4NF.

- nicht-trivial $C \not\subseteq AB$ und $\gamma \neq \emptyset$

aber AB ~~kein~~ ist Superschlüssel

[UPDATE]

Höchste NF bestimmen

e) $\mathcal{R} = \{A, B, C, D, E\}$, $F_{\mathcal{R}} = \{\overset{\alpha}{A} \rightarrow BCDE\}$, $D_{\mathcal{R}} = \{\underbrace{BC \twoheadrightarrow E}_{1}\}$

1NF ✓
2NF ✓
3NF ✓
BCNF ✓

4NF ✗ MVD (1) verletzt 4NF
- linke Seite {B,C} ist kein Superschlüssel in $\mathcal{R}$
- und sie ist nicht trivial $E \not\subseteq BC$ und $\gamma \neq \emptyset$

f) $\mathcal{R} = \{A, B, C, D, E\}$, $F_{\mathcal{R}} = \{A \rightarrow BCDE\}$, $D_{\mathcal{R}} = \{AB \twoheadrightarrow E\}$

AB ist Superschlüssel in $\mathcal{R}$

1NF - BCNF wie in e)

4NF ✓

Schema in 3NF bringen

Synthesealgorithmus

- Eingabe:
  - **Kanonische Überdeckung $\mathcal{F}_c$**
    - Linksreduktion
    - Rechtsreduktion
    - FDs der Form $\alpha \to \emptyset$ (sofern vorhanden)
    - FDs mit gleicher linke Seite zusammenfassen
- Algorithmus:
  1. Für jede FD $\alpha \to \beta$ in $\mathcal{F}_c$ forme ein Unterschema $\mathcal{R}_\alpha = \alpha \cup \beta$, ordne $\mathcal{R}_\alpha$ die FDs $\mathcal{F}_\alpha := \{\alpha' \to \beta' \in \mathcal{F}_c \mid \alpha' \cup \beta' \subseteq \mathcal{R}_\alpha\}$ zu
  2. Füge ein Schema $\mathcal{R}_\kappa$ mit einem Kandidatenschlüssel hinzu
  3. Eliminiere redundante Schemata, d.h. falls $\mathcal{R}_i \subseteq \mathcal{R}_j$, verwerfe $\mathcal{R}_i$

Synthesealgorithmus

$$\mathcal{R} = \{A, B, C, D, E, F, G\}, F_{\mathcal{R}} = \{A \to \cancel{B}CDE,\ E \to FB,\ F \to A\}$$

Kan. Überdeckung

1) Linksreduktion: ✓

2) Rechtsreduktion: betrachte FD

mit FD 1 und 2

$$\underbrace{\bullet\ A \to \cancel{B}CDE \qquad \bullet\ E \to FB \qquad \bullet\ F \to A}_{F_c}$$

3) entfernen von $\alpha \to \emptyset$: keine

4) zusammenfassen: keine gleichen α's

Synthese:

- R_1: {A, C, D, E} R_2: {E, F, B} R_3: {F, A}
- Kandidatenschlüssel {A, G} G ist nicht abhängig von den anderen Attributen => Teil des Kandidatenschlüssels

kein R_i enthält den KS

=> R_4: {A, G}

Schema in BCNF bringen

BCNF-Dekompositionsalgorithmus

- Starte mit $Z = \{\mathcal{R}\}$
- Solange es noch ein $\mathcal{R}_i \in Z$ gibt, das nicht in BCNF ist:
  - Finde eine FD $(\alpha \to \beta) \in F^+$ mit
    - $\alpha \cup \beta \subseteq \mathcal{R}_i$
    - $\alpha \cap \beta = \emptyset$
    - $\alpha \to \mathcal{R}_i \notin F^+$
  - Zerlege $\mathcal{R}_i$ in $\mathcal{R}_{i.1} := \alpha \cup \beta$ und $\mathcal{R}_{i.2} := \mathcal{R}_i - \beta$
  - Entferne $\mathcal{R}_i$ aus Z und füge $\mathcal{R}_{i.1}$ und $\mathcal{R}_{i.2}$ ein, also $Z := (Z - \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i.1}\} \cup \{\mathcal{R}_{i.2}\}$

Schema in 4NF bringen

4NF-Dekompositionsalgorithmus

- Starte mit $Z = \{\mathcal{R}\}$
- Solange es noch ein $\mathcal{R}_i \in Z$ gibt, das nicht in 4NF ist:
  - Finde eine MVD $\alpha \twoheadrightarrow \beta \in \mathcal{F}^+$ mit
    - $\alpha \cup \beta \subset \mathcal{R}_i$
    - $\alpha \cap \beta = \emptyset$
    - $\alpha \rightarrow \mathcal{R}_i \notin \mathcal{F}^+$
  - Zerlege $\mathcal{R}_i$ in $\mathcal{R}_{i.1} := \alpha \cup \beta$ und $\mathcal{R}_{i.2} := \mathcal{R}_i - \beta$
  - Entferne $\mathcal{R}_i$ aus Z und füge $\mathcal{R}_{i.1}$ und $\mathcal{R}_{i.2}$ ein, also $Z := (Z - \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i.1}\} \cup \{\mathcal{R}_{i.2}\}$

BCNF-Dekompositionsalgorithmus

$$\mathcal{R} = \{A, B, C, D, E, F\},\ F_{\mathcal{R}} = \{B \to AD,\ DEF \to B,\ C \to AE\}$$

- Finde eine FD, die die BCNF verletzt:

  $B \to AD$ $\{B\} \cap \{AD\} = \emptyset$ aber B kein Superschlüssel

  Zerlege in $R_1: \{\underline{B}, A, D\}$ und $R_2: \{B, C, E, F\}$

  FD 2 geht verloren
  FD 3 teilweise

- FD 2' in R_2 verletzt BCNF

  zerlege R_2 in $R_{2.1}: \{\underline{C}, E\}$ und $R_{2.2}: \{\underline{C, B, F}\}$

  ↳ trivialer Superschlüssel, da nur noch triviale FDs gelten.

SQL (1)

Finde die “Wechsler”.

Ausgabe: Name, MatrNr, Anzahl Besuchte Gruppen (absteigend sortiert).

Übungen			
MatrNr	Gruppe	Punkte	Woche
24002	1	1	1
24002	1	0	2
26120	2	0	1
26120	3	1	2
...	...	...	...

```
with
Uebungen(MatrNr,Gruppe,Punkte,Woche) as (
values
  (24002,1,1,1),
  (24002,1,0,2),
  (26120,2,0,1),
  (26120,3,1,2)
)
select s.name, s.matrnr, count(DISTINCT gruppe) besucht
from uebungen u, studenten s
where u.matrnr = s.matrnr
group by s.matrnr, s.name
order by besucht desc
```

SQL (2)

Finde Student(en) mit höchster Punktzahl.

Ausgabe: Name, MatrNr, Punktestand ~~(absteigend sortiert)~~.

Übungen			
MatrNr	Gruppe	Punkte	Woche
24002	1	1	1
24002	1	0	2
26120	2	0	1
26120	3	1	2
...	...	...	...

```
with punkte as (
  select u.matrnr, sum(u.punkte) as sumpunkte
  from uebungen u
  group by u.matrnr
)
select s.name, s.matrnr
from punkte p, studenten s
where p.matrnr = s.matrnr
  and p.sumpunkte = (select max(sumpunkte) from uebungen)
```

alternativ mit NOT EXISTS (

```
with punkte as (
  select u.matrnr, sum(u.punkte) as sumpunkte
  from uebungen u
  group by u.matrnr
)
select s.name, s.matrnr
from punkte p, studenten s
where p.matrnr = s.matrnr
  and not exists (
    select * from punkte p2
    where p2.sumpunkte > p.sumpunkte)
```

muss leer sein

SQL (3)

Bonus verrechnen (wenn Anzahl Punkte > 1).

Ausgabe: Name, MatrNr, '[Kein] Bonus erhalten'.

Übungen			
MatrNr	Gruppe	Punkte	Woche
24002	1	1	1
24002	1	0	2
26120	2	0	1
26120	3	1	2
...	...	...	...

```
with punkte as (
  select u.matrnr, sum(u.punkte) as sumpunkte
  from uebungen u
  group by u.matrnr
)
select s.name, s.matrnr,
   case when p.sumpunkte > 1 then 'Bonus erhalten'
        else 'Kein Bonus erhalten'
   end as bonus
from punkte p, studenten s
where p.matrnr = s.matrnr
```