Grundlagen: Datenbanken

Zentralübung / Wiederholung / Fragestunde

Moritz Sichert Lukas Vogel

gdb@in.tum.de

WiSe 2018 / 2019

Plattform für Fragen

https://arsnova.eu/mobile/#id/65746822

Diese Folien finden Sie online.

Die Mitschrift stellen wir im Anschluss online.

Agenda

- Hinweise zur Klausur
- Stoffübersicht/-Diskussion
- Wiederholung + Übung
  - Datenbankentwurf
  - Relationale Algebra
  - Tupel- / Domänenkalkül
  - SQL
  - Relationale Entwurfstheorie
  - Erweiterbares Hashing
  - B-Baum
  - R-Baum
  - Anfrageoptimierung

Hinweise zur Klausur

Termine

- 1. Klausurtermin - **Mi. 13.02.2019, 10:30 bis 12:00 Uhr**
- Notenbekanntgabe / Anmeldung zu Einsicht - **vsl. Fr. 29.02.2019 (ab Mittag)**
- Einsicht - **Mo. 04.03.2019**
- Anmeldung zur 2. Klausur von - **18.03.2019**, bis **01.04.2019**
- 2. Klausurtermin - **TBA**

Verschiedenes

- Wenn Sie nicht zur Klausur kommen: **Bitte abmelden!**
- **Raumbekanntgabe**, via TUMonline sowie in Moodle
- 90 Minuten / 90 Punkte
- Sitzplatzvergabe (Aushang: MatrNr $\rightarrow$ Sitzplatz, KEINE Namensnennung)
- Einsichtnahme (Instruktionen in Moodle, nach Notenbekanntgabe)
- Bonus: Gilt für beide Klausuren.

Stoffübersicht (1)

Datenbankentwurf / ER-Modellierung

- ER-Diagramme, Funktionalitäten, Min-Max, Übersetzung ER $\leftrightarrow$ Relational, Schemavereinfachung/-verfeinerung

Das Relationale Modell

- Stichworte: Schema, Instanz/Ausprägung, Tupel, Attribute, ...
- Anfragesprachen
  - Relationale Algebra
    - RA-Operatoren: Projektion, Selektion, Join (Theta, Natural, Outer, Semi, Anti), Kreuzprodukt, Mengendifferenz/-vereinigung/-schnitt, Division
  - Tupelkalkül, ~~Domänenkalkül~~

Stoffübersicht (2)

SQL

- DDL
  - Create/~~Alter/Drop~~ Table
  - Integritätsbedingungen: primary key (auch zusammengesetzt), not null, foreign key, on delete (cascade/set null), check constraints, ...
  - Insert, Update, Delete
- Queries
  - Select/From/Where
  - Joins: inner, (left/right/full) outer
  - Sortieren: Order by (asc/desc)
  - Aggregation: Group by, having, sum(), avg(), max(), ...
  - Geschachtelte Anfragen
  - Quantifizierte Unteranfragen: where (not) exists
  - Mengenoperatoren: union, intersect, except (all)
  - Modularisierung: with x as ...
  - Spezielle Sprachkonstrukte: between, like, case when. ...
  - **Rekursion!**

Stoffübersicht (3)

Relationale Entwurfstheorie

- Definitionen:
  - Funktionale Abhängigkeiten (FDs), Armstrong-Axiome (+Regeln), FD-Hülle, Kanonische Überdeckung, Attribut-Hülle, Kandidaten-/Superschlüssel, Mehrwertige Abhängigkeiten (MVDs), Komplementregel, Triviale FDs/MVDs,...
- Normalformen: 1., 2., 3.NF, BCNF und 4. NF
- Zerlegung von Relationen
  - in 3.NF mit dem Synthesealgorithmus
  - in BCNF/4.NF (zwei Varianten des Dekompositionsalgorithmus)
  - Stichworte: Verlustlos, Abhängigkeitsbewahrend

Stoffübersicht (4)

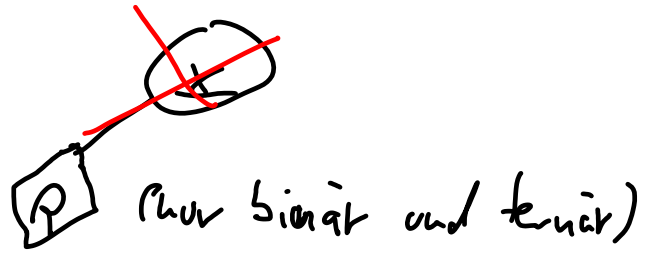
- **Physische Datenorganisation**
  - Speicherhierarchie
  - HDD/RAID 4, 5, 6
  - TID-Konzept
  - Indexstrukturen
    - B-Baum, B+-Baum
    - R-Baum
    - Erweiterbares Hashing
- **Anfragebearbeitung**
  - Kanonische Übersetzung (SQL $\rightarrow$ Relationale Algebra)
  - Logische Optimierung (in relationaler Algebra)
    - Frühzeitige Selektion, Kreuzprodukte $\rightarrow$ Joins, Joinreihenfolge
  - Implementierung relationaler Operatoren
    - Nested-Loop-Join
    - Sort-Merge-Join
    - Hash-Join
    - Index-Join

Stoffübersicht (5)

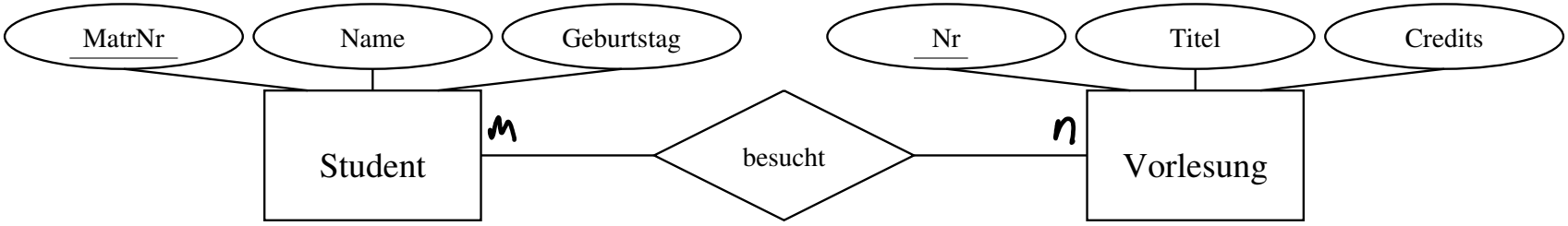
- **Transaktionsverwaltung**
  - BOT, read, write, commit, abort
  - Rollback (R1 Recovery)
  - ACID-Eigenschaften
- **Fehlerbehandlung (Recovery)**
  - Fehlerklassifikation (R1 - R4)
  - Protokollierung: Redo/Undo, physisch/logisch, Before/After-Image, WAL, LSN
  - Pufferverwaltung: Seite, FIX, Ersetzungsstrategie steal/$\neg$steal, Einbringstrategie force/$\neg$force
  - Wiederanlauf nach Fehler, Fehlertoleranz des Wiederanlaufs, Sicherungspunkte
- **Mehrbenutzersynchronisation**
  - Formale Definition einer Transaktion (TA)
  - Historien (Schedules)
    - Konfliktoperationen, (Konflikt-)Äquivalenz, Eigenschaften von Historien
  - Datenbank-Scheduler
    - pessimistisch (sperrbasiert, zeitstempelbasiert), optimistisch

Datenbankentwurf

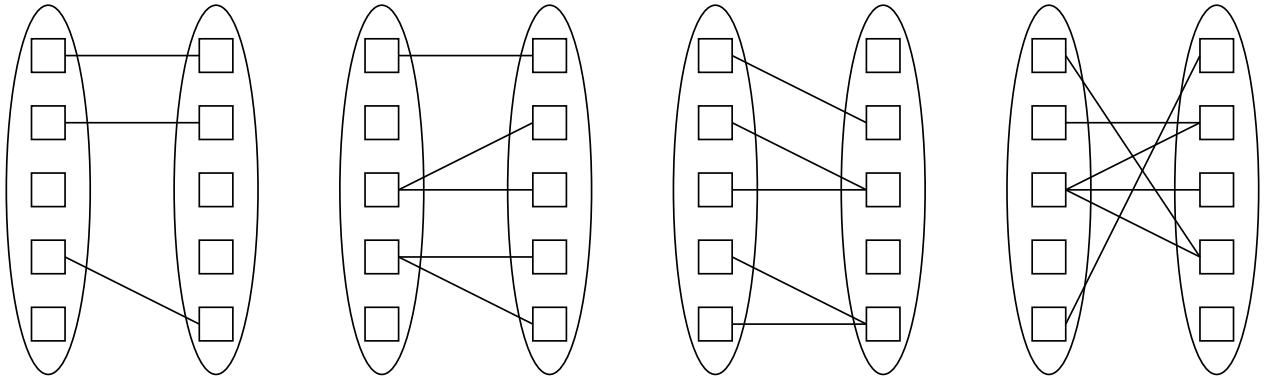
- Entity-Relationship-Diagramme:
  - Entities
  - Relationships zwischen Entities
  - Attribute
  - Schlüssel nur bei Entities
- Funktionalitätsangaben, Min-Max-Angaben
- Überführung in relationales Schema:
  - Eine Relation für jede Entity
  - Eine Relation für jede Relationship, Schlüsselattribute der Entitites übernehmen
  - Verfeinerung: Relationen zusammenfassen, die den selben Schlüssel haben (1:N, N:1, 1:1)



Datenbankentwurf

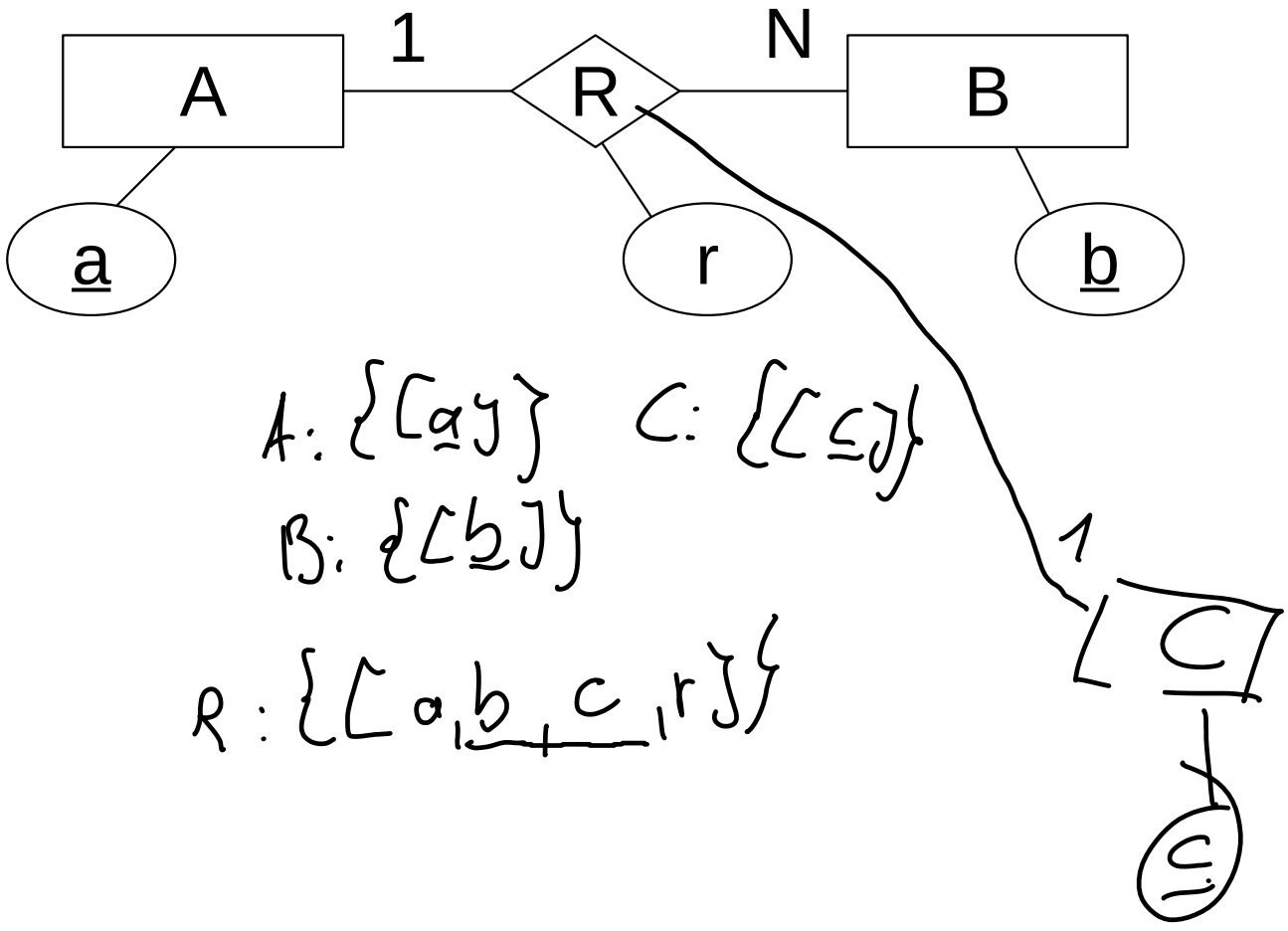


Student: {[MatrNr, Name, GebTag]}
Vorlesung: {[Nr, Titel, Creditpts]}
besucht: {[MatrNr, Nr]}

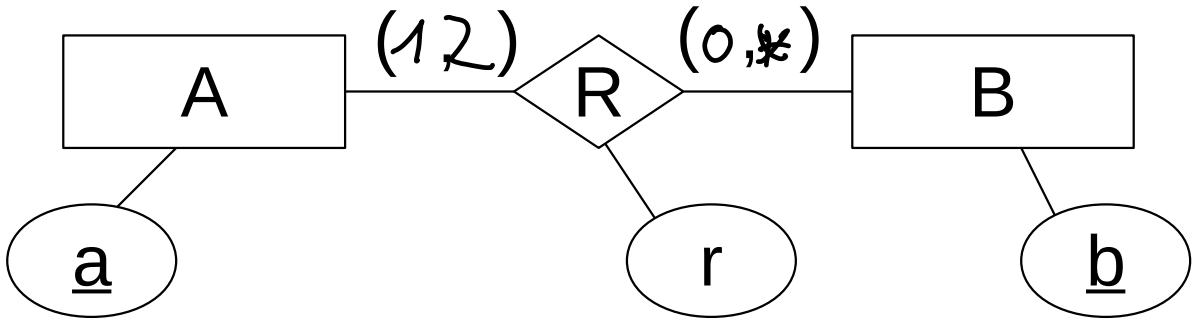


1:1 1:N N:1 M:N

ER-Modell in Schema überführen und verfeinern



(Min,Max) - Angaben



R		
a	b	r
1	2	x
1	3	y
2	4	z

Relationale Algebra

Algebraische Operatoren:

Projektion	$\Pi_{A_1,\ldots,A_n}$
Selektion	σ_p
Kreuzprodukt	$\times$
Verbund (Join)	$\bowtie_\theta$, $⟗_\theta$, $⟕_\theta$, $⟖_\theta$, $\ltimes_\theta$, $\rtimes_\theta$, $▷_\theta$, $◁_\theta$
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 $\rho_{a_1\leftarrow b_1,\ldots,a_n\leftarrow b_n}$

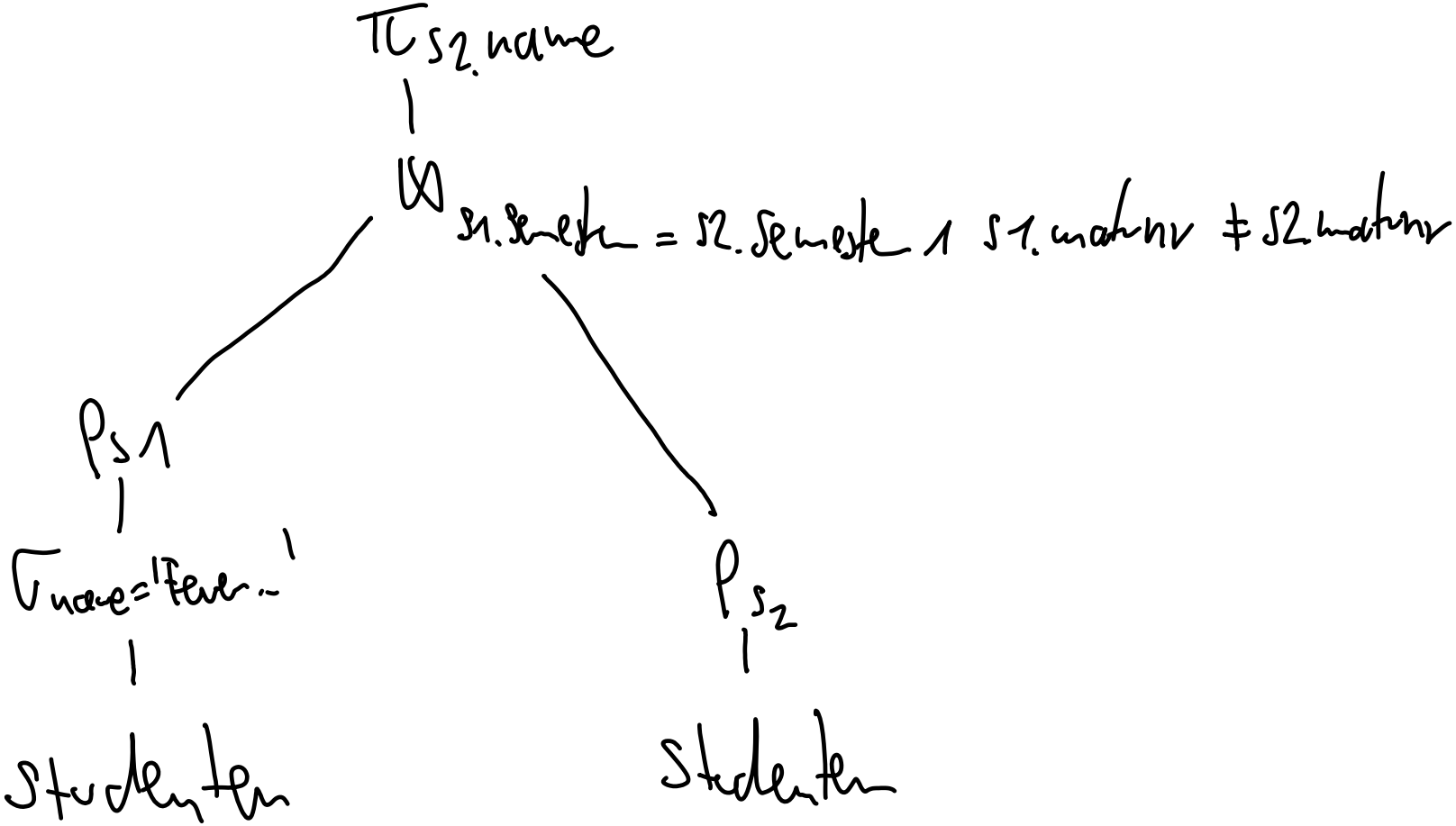
Anmerkung: Natural-Join vs. allgemeiner Theta-Join

	Natural	Theta
Inner	$\bowtie$	$\bowtie_\theta$
Outer	$⟗$, $⟕$, $⟖$	$⟗_\theta$, $⟕_\theta$, $⟖_\theta$
Semi	$\ltimes$, $\rtimes$	$\ltimes_\theta$, $\rtimes_\theta$
Anti	$\triangleright$, $\triangleleft$	$\triangleright_\theta$, $\triangleleft_\theta$

- Natural
  - Implizite Gleichheitsbedingung auf gleichnamigen Attributen
  - Die gleichnamigen Attribute tauchen im Ergebnis nur einmal auf (inner und outer).
- Theta
  - Explizite (beliebige) Joinbedingung: θ.
  - Im Falle von Inner- und Outer-Join werden alle Attribute der beiden Eingaberelationen in das Ergebnis projiziert.

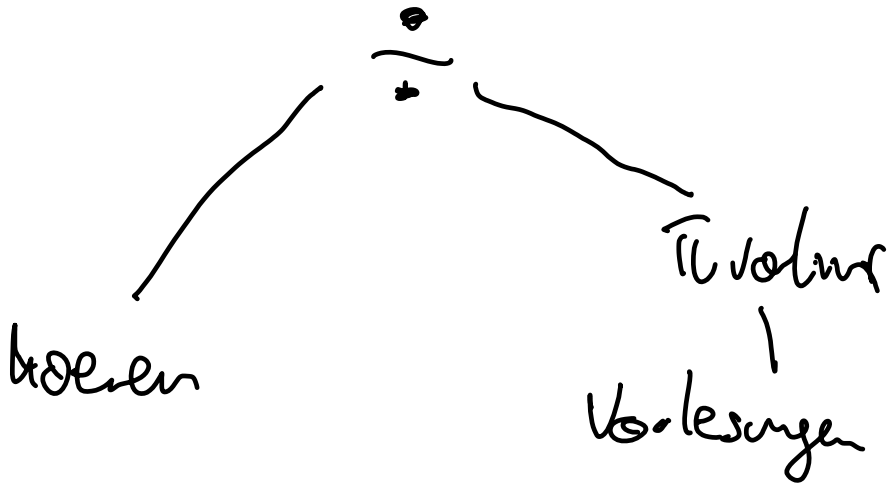
Übung: Relationale Algebra (1)

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



Übung: Relationale Algebra (2)

Finde Studenten (nur MatrNr ausgeben), die alle Vorlesungen gehört haben.



Tupel- / ~~Domänen~~kalkül

- Schreibweise Tupelkalkül: $\{\, t \mid p(t) \,\}$ bzw. $\{\, [t.a1, t.a2] \mid p(t) \,\}$
- Schreibweise Domänenkalkül: $\{\, [a1, a2, a3] \mid p(a1, a2, a3) \,\}$
- Neue Variablen in Prädikat ausschließlich erzeugt durch Quantoren ($\exists, \forall$)
- Relationen können als Mengen im Prädikat verwendet werden, z.B. $s \in$ Studenten bzw. $[m, v] \in$ hoeren

Übung: Tupel- / Domänenkalkül

Finden Sie Studierende, die noch keine Vorlesung gehört haben.

$\{ s \mid s \in Studenten \wedge \neg(\exists h \in hören ($

$s.matrnr = h.matrnr$

$)) \}$

$\ldots \quad \forall s \in Studenten ($

$s.semester = 3 \Rightarrow (\ldots)$

$)$

$$\{\, s \mid s \in \mathsf{Studenten} \wedge \neg \exists h \in \mathsf{hoeren}(h.\mathsf{matrnr} = s.\mathsf{matrnr}) \,\}$$

SQL

- Relationen erzeugen:

```
create table X ( a integer primary key, ... )
```

- Werte einfügen:

```
insert into X values (1) / select ...
```

- Form einer Query:

```
with X as (...)
select ...
from ...
where ...
group by ...
having ...
order by ...
union/intersect/except (all)
select ...
```

Übung: SQL (DDL I)

Geben Sie ein SQL-Statement an, das die Relation „prüfen“ erzeugt.

```
create table pruefen(                                on delete/update ...
    matrnr integer not null references Studenten(matrnr),
    vorlnr integer not null,
    persnr integer not null,
    note decimal(2,1) not null,
    primary key (matrnr, vorlnr)
)
```

```
create table pruefen (
    matrnr integer not null
        references studenten (matrnr)
        on update cascade on delete cascade,
    vorlnr integer not null
        references vorlesungen (vorlnr)
        on update cascade,
    persnr integer not null
        references professoren (persnr)
        on update cascade,
    note decimal(2, 1) not null,
    primary key (matrnr, vorlnr),
    check (note >= 1.0 and note <= 5.0)
)
```

Übung: SQL (DDL II)

Schreiben Sie ein SQL-Statement, das für alle Studenten, die die Vorlesung „GDB“ hören, eine Prüfung bei Prüfer „Kemper“ mit der Note 1,0 einträgt.

insert into pruefen

Select matrnr, vorlnr, persnr, 1.0 as note

from ...

```
insert into pruefen
select h.matrnr, v.vorlnr, p.persnr, 1.0
from hoeren h, vorlesungen v, professoren p
where
    h.vorlnr = v.vorlnr and
    v.titel = 'GDB' and
    p.name = 'Kemper'
```

Übung: SQL (Aggregation)

Wieviele Studierende gibt es pro Semester?

```
select semester, count(*) as AnzStud
from Studenten
group by Semester
having count(*) > 1
```

```
select s.semester, count(*)
from studenten s
group by s.semester
```

Übung: SQL (Aggregation / Allquantor)

$\neg \exists s \in S : p(s) \Leftrightarrow \forall s \in S : \neg p(s)$

Finden Sie alle Studenten, die alle Vorlesungen hören. Geben Sie zwei Lösungen an: Eine, die auf Zählen basiert, und eine, die Quantoren verwendet.

$\neg\neg \forall v : p(v) \Leftrightarrow \neg(\exists v : \neg p(v))$

```
select * from studenten s
where not exists (
        select * from vorlesungen v
        where not exists (
                select * from hoeren
                where h.matrnr = s.matrnr and
                      h.vorlnr = v.vorlnr
                )
)
```

```sql
select s.*
from studenten s
where
    (select count(*) from hoeren h where h.matrnr = s.matrnr) =
    (select count(*) from Vorlesungen)
```

```
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
            h.vorlnr = v.vorlnr
    )
)
```

Übung: SQL (Rekursion I)

Geben Sie alle Titel der (rekursiven) Voraussetzungen der Vorlesung „Bioethik“ aus.

```
with recursive voraussetzen_rec as (
    select vorlnr
    from vorlesungen v
    where titel = 'Bioethik'
    union all
    select vor.vorgaenger
    from voraussetzen_rec v, voraussetzen vor
    where v.vorlnr = vor.nachfolger
)
select v.titel
from vorlesungen v, voraussetzen_rec vor
where
    v.vorlnr = vor.vorlnr and
    v.titel != 'Bioethik'
```

Übung: SQL (Rekursion II)

Geben Sie das früheste Semester an, in dem man Bioethik hören kann, wenn man alle (rekursiven) voraussetzende Vorlesungen hört.

```
with recursive vor_rec (
base  select v.vorlnr, 1 as semester
case  from Vorlesungen v
      where v.Titel = 'Bioethik'

    union all(

Rekursions-   select vor.vorgaenger, v.semester+1
schritt       from vor_rec v, voraussetzen vor
              where v.vorlnr = vor.nachfolger
    )

select max(semester) from vor_rec
```

```
with recursive voraussetzen_rec as (
    select vorlnr, 1 as semester
    from vorlesungen v
    where titel = 'Bioethik'
    union all
    select vor.vorgaenger, v.semester + 1
    from voraussetzen_rec v, voraussetzen vor
    where v.vorlnr = vor.nachfolger
)
select max(semester) from voraussetzen_rec
```

Übung: SQL (Rekursion III)

Finden Sie alle Studenten, die Kant direkt oder indirekt kennen.

Ein Student kennt Kant direkt, wenn er eine seiner Vorlesungen besucht. Ein Student kennt Kant indirekt, wenn er eine Vorlesung besucht, die auch ein anderer Student hört, der Kant kennt (direkt oder indirekt).

```
with recursive stud_von_kant as (
    select distinct h.matrnr
    from hoeren h, vorlesungen v, professoren p
    where
        h.vorlnr = v.vorlnr and
        v.gelesenVon = p.persnr and
        p.name = 'Kant'
),
studenten_kennen as (
    select * from stud_von_kant
    union
    select h2.matrnr
    from studenten_kennen s, hoeren h1, hoeren h2
    where
        s.matrnr = h1.matrnr and
        h1.vorlnr = h2.vorlnr
)
select s.*
from studenten_kennen sk, studenten s
where sk.matrnr = s.matrnr
```

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 $r.\beta = t.\beta$.

Übung: Relationenausprägung vervollständigen

Gegen seien die folgende Relationenausprägung und die funktionalen Abhängigkeiten. Bestimmen Sie zunächst x und danach y, sodass die FDs gelten.

$$B \rightarrow A$$
$$AC \rightarrow D$$

A	B	C	D
7	3	5	8
x 1	4	2	8
7	3	6	9
1	4	2	y 8
1	5	7	7 10

Funktionale Abhängigkeiten

Seien α,β,γ,$\delta \subseteq \mathcal{R}$

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

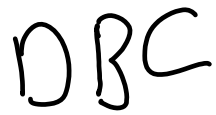
- Schlüssel identifizieren jedes Tupel einer Relation $\mathcal{R}$ eindeutig.
- Eine Attributmenge $\alpha \subseteq \mathcal{R}$ ist ein **Superschlüssel**, gdw. $\alpha \to \mathcal{R}$
- Ist α zudem noch minimal, ist es auch ein **Kandidatenschlüssel** (meist mit κ bezeichnet)
  - Es existiert also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

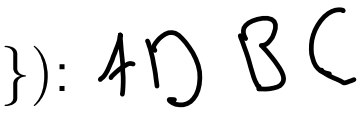
- I.A. existieren mehrere Super- und Kandidatenschlüssel..
- Man muss sich bei der Realisierung für einen Kandidatenschlüssel entscheiden, dieser wird dann **Primärschlüssel** genannt.
- Der triviale Schlüssel $\alpha = \mathcal{R}$ existiert immer.

Übung: Schlüsseleigenschaft von Attributmengen ermitteln

- Ob ein gegebenes α ein Schlüssel ist, kann mithilfe der Armstrong-Axiome ermittelt werden
- Besser: Die **Attributhülle** $AH(\alpha)$ bestimmen.

- Beispiel: $\mathcal{R} = \{ A , B , C , D \}$, mit $F_{\mathcal{R}} = \{AB \to CD, B \to C, D \to B\}$

$AH(\{D\})$: 

$AH(\{A, D\})$: 

$AH(\{A, B, D\})$: