# Grundlagen: Datenbanken

## Zentralübung / Wiederholung / Fragestunde

Michael Jungmair Stefan Lehner Christoph Anneser

gdb@in.tum.de

WiSe 2023 / 2024

Diese Folien finden Sie online.

# Agenda

- Hinweise zur Klausur
- Stoffübersicht/-Diskussion
- Wiederholung + Übung
  1. Tupel- und Domänenkalkül
  2. Relationale Entwurfstheorie: FDs, MVDs, Normalformen
  3. SQL + Rekursion
  4. Relationale Algebra

# Hinweise zur Klausur (1)

**Termine**

- 1. Klausurtermin - **Fr 16.02.2024, 17:00 bis 18:30 Uhr**
- Notenbekanntgabe - **vsl. Mo. 03.03.2024**
- Einsicht - **vsl. 03.-06.03.2024 (online)**
- 2. Klausurtermin - vsl. zwischen **01.04.2024** und **15.04.2024**
- Wenn Sie nicht zur Klausur kommen: **Bitte abmelden!**

**Räume und Sitzplätze**

- Raumbekanntgabe via TUMonline sowie in Moodle (Zuordnung Matrikelnummer zu Hörsaal)
- **Achtung**: Manche Hörsäle im **Campus München Innenstadt** (Arcisstraße)!
- Individuelle Sitzplatzzuteilung in TUMonline einsehbar

# Hinweise zur Klausur (2)

**Sonstiges**

- 90 Minuten / 90 Punkte
- Hilfsmittel (Notizzettel, Taschenrechner, etc.) nicht erlaubt, unbeschriftetes Wörterbuch Deutsch $\leftrightarrow$ Fremdsprache erlaubt
- Bonus: Gilt für beide Klausuren
- Sollten Sie einen Nachteilsausgleich genehmigt bekommen haben, melden Sie sich rechtzeitig bei uns!

# 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, Aggregation
  - Tupelkalkül, Domänenkalkül

# Stoffübersicht (2)

**SQL**

- DDL
  - Create/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, coalesce ...
  - 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
  - RAID
  - Indexstrukturen
    - B-Baum, B+-Baum
    - Erweiterbares Hashing
    - R-Baum

- **Anfragebearbeitung**
  - Kanonische Übersetzung (SQL $\rightarrow$ Relationale Algebra)
  - Logische Optimierung (in relationaler Algebra)
    - Frühzeitige Selektion, Kreuzprodukte $\rightarrow$ Joins, Joinreihenfolge
  - Implementierung physischer Operatoren (Iteratormodell)
    - 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, steal/$\neg$steal, force/$\neg$force
  - Wiederanlauf nach Fehler, Fehlertoleranz des Wiederanlaufs, Sicherungspunkte

# Stoffübersicht (6)

- **Mehrbenutzersynchronisation**
  - Formale Definition einer Transaktion (TA)
  - Historien (Schedules)
    - Konfliktoperationen, (Konflikt-)Äquivalenz, Eigenschaften von Historien
  - Datenbank-Scheduler
    - sperrbasiert, X-Sperren, S-Sperren
    - (striktes) 2PL
- **Sicherheitsaspekte**
  - SQL-Injections

# 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 (all)
select ...
```

# Übung: SQL (Rekursion)

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 (DDL)

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

```
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 (DML)

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 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'
```

# Relationale Algebra

**Algebraische Operatoren:**

| | |
|---|---|
| Projektion | $\Pi_{A_1,\ldots,A_n}$ |
| Selektion | $\sigma_p$ |
| Kreuzprodukt | $\times$ |
| Verbund (Join) | $\bowtie_\theta$, $\rtimes_\theta$, $\ltimes_\theta$, $⟗_\theta$, $⟕_\theta$, $⟖_\theta$, $\triangleright_\theta$, $\triangleleft_\theta$ |
| Mengenoperationen | $\cup$, $\cap$, $\setminus$ |
| Division | $\div$ |
| Gruppierung/Aggregation | $\Gamma_{A_1,\ldots,A_n\,;\,a_1:f_1,\ldots,a_m:f_m}$ |
| Umbenennung | $\rho_N$, 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: $\theta$.
  - 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- / Domänenkalkül

- Schreibweise Tupelkalkül: $\{ t \mid p(t) \}$ bzw. $\{ [t.a1, t.a2] \mid p(t) \}$
- Schreibweise Domänenkalkül: $\{ [a1, a2, a3] \mid p(a1, a2, a3) \}$
- $p$ ist eine Formel
  - Atome (vergleiche mit Attributen) sind Formeln
  - Formeln können verbunden werden mit aussagenlogischen Prädikaten ($\wedge, \vee, \neg, \Rightarrow$)
- 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 (1)

Finden Sie Studierende, die noch keine Vorlesung gehört haben.

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

# Übung: Tupel- / Domänenkalkül (2)

Finden Sie Studierende, die alle Vorlesung mit mehr als 4 SWS gehört haben.

# Relationale Entwurftheorie

**Funktionale Abhängigkeiten** (kurz FDs, für functional dependencies):

- Seien $\alpha$ und $\beta$ 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.

$$\begin{aligned} B &\rightarrow A \\ AC &\rightarrow D \end{aligned}$$

| A | B | C | D |
|---|---|---|---|
| 7 | 3 | 5 | 8 |
| $x$ | 4 | 2 | 8 |
| 7 | 3 | 6 | 9 |
| 1 | 4 | 2 | $y$ |

# Funktionale Abhängigkeiten

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

**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. $\alpha \rightarrow \mathcal{R}$
- Ist $\alpha$ zudem noch minimal, ist es auch ein **Kandidatenschlüssel** (meist mit $\kappa$ bezeichnet)
  - Es existiert also kein $\alpha' \subset \alpha$ für das gilt: $\alpha' \rightarrow \mathcal{R}$

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

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

$AH(\{D\})$:

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

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

# Mehrwertige Abhängigkeiten

## multivalued dependencies (MVDs)

"Halb-formal":

- Seien $\alpha$ und $\beta$ disjunkte Teilmengen von $\mathcal{R}$
- und $\gamma = (\mathcal{R} \backslash \alpha) \backslash \beta$
- dann ist $\beta$ mehrwertig abhängig von $\alpha$ ($\alpha \twoheadrightarrow \beta$), wenn in jeder gültigen Ausprägung von $\mathcal{R}$ gilt:
- Bei zwei Tupeln mit gleichem $\alpha$-Wert kann man die $\beta$-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 $\gamma = \emptyset$)

# Beispiel: Mehrwertige Abhängigkeiten

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

| ProfessorIn | Vorlesung | AssistentIn |
|---|---|---|
| K | GDB | Linnea |
| K | WebDB | Linnea |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

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

- **1. NF**: Attribute haben nur atomare Werte, sind also nicht mengenwertig.
- **2. NF**: Jedes Nichtschlüsselattribut ist voll funktional abhängig von jedem Kandidatenschlüssel.
  - $\beta$ hängt **voll funktional** von $\alpha$ ab ($\alpha \xrightarrow{\bullet} \beta$), gdw. $\alpha \rightarrow \beta$ und es existiert kein $\alpha' \subset \alpha$, so dass $\alpha' \rightarrow \beta$ gilt.
- **3. NF**: Für alle geltenden nicht-trivialen FDs $\alpha \rightarrow \beta$ gilt entweder
  - $\alpha$ ist ein Superschlüssel, oder
  - jedes Attribut in $\beta$ ist in einem Kandidatenschlüssel enthalten
- **BCNF**: Die linken Seiten $(\alpha)$ aller geltenden nicht-trivalen FDs sind Superschlüssel.
- **4. NF**: Die linken Seiten $(\alpha)$ aller geltenden nicht-trivalen MVDs sind Superschlüssel.

# Übung: Höchste NF bestimmen

$\mathcal{R} : \{[\ A\,, B\,, C\,, D\,, E\ ]\}$

$$\begin{aligned} A &\rightarrow BCDE \\ AB &\rightarrow C \end{aligned}$$

- ○ 1. NF
- ○ 2. NF
- ○ 3. NF
- ○ BCNF
- ○ 4. NF
- ○ keine der angegebenen

# Übung: Höchste NF bestimmen (2)

$\mathcal{R} : \{[\ A\,,\ B\,,\ C\,,\ D\,,\ E\ ]\}$

$$A \rightarrow BCDE$$
$$B \rightarrow C$$

- ○ 1. NF
- ○ 2. NF
- ○ 3. NF
- ○ BCNF
- ○ 4. NF
- ○ keine der angegebenen

# Schema in 3. NF überführen

## Synthesealgorithmus

- Eingabe:
  - **Kanonische Überdeckung** $\mathcal{F}_c$
    - Linksreduktion
    - Rechtsreduktion
    - FDs der Form $\alpha \to \emptyset$ entfernen (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 $\mathcal{R}_i$
- Ausgabe:
  - Eine Zerlegung des unsprünglichen Schemas, in der alle Schemata in 3.NF sind.
  - Die Zerlegung ist **abhängigkeitsbewahrend** und **verlustlos**.

# Übung: Synthesealgorithmus

$$\mathcal{R} : \{[\ A\,,\, B\,,\, C\,,\, D\,,\, E\,,\, F\ ]\}$$

$$\begin{aligned} B &\rightarrow ACDEF \\ EF &\rightarrow BC \\ A &\rightarrow D \end{aligned}$$

# Schema in BCNF überführen

**BCNF-Dekompositionsalgorithmus** (nicht abhängigkeitsbewahrend)

- 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
    - $\alpha \cup \beta \subseteq \mathcal{R}_i$ (FD muss in $\mathcal{R}_i$ gelten)
    - $\alpha \cap \beta = \emptyset$ (linke und rechte Seite sind disjunkt)
    - $\alpha \to \mathcal{R}_i \notin F^+$ (linke Seite ist kein Superschlüssel)
  - 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 4.NF überführen

**4NF-Dekompositionsalgorithmus** (nicht abhängigkeitsbewahrend)

- 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
    - $\alpha \cup \beta \subset \mathcal{R}_i$ (FD muss in $\mathcal{R}_i$ gelten und nicht-trivial sein)
    - $\alpha \cap \beta = \emptyset$ (linke und rechte Seite sind disjunkt)
    - $\alpha \rightarrow \mathcal{R}_i \notin \mathcal{F}^+$ (linke Seite ist kein Superschlüssel)
  - 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 $Z := (Z - \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i.1}\} \cup \{\mathcal{R}_{i.2}\}$

# Übung: BCNF-Dekompositionsalgorithmus

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

Wir wünschen viel Erfolg. :-)