Grundlagen: Datenbanken WS 15/16

2. Zentralübung / Wiederholung / Fragestunde

Harald Lang

gdb@in.tum.de

Diese Folien finden Sie online.

Die Mitschrift erhalten Sie im Anschluss.

Termine

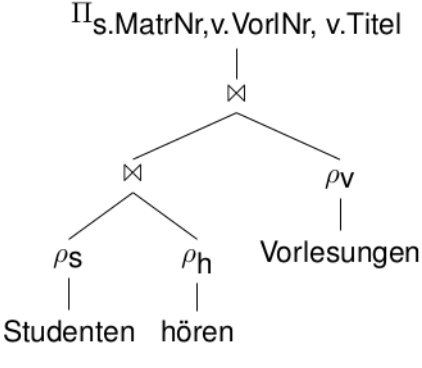
- Klausur
  - 24.02.2016, 10:30 - 12:30 Uhr
  - Anmeldung bis 15.01.2016
- Weihnachtsferien
  - 24.12.2015 - 6.1.2016
  - HINWEIS: **Ein** Übungblatt für die Wochen 21.-23.12 (Mo-Mi) und 7.-8.1. (Do, Fr).

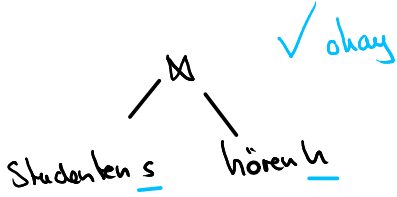
Agenda

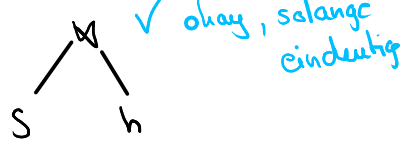
- Relationale Anfragesprachen
  - Anmerkungen zur relationalen Algebra
  - Anmerkung: Natural-Join vs. allgemeiner Theta-Join
  - Kalkül: Freie vs. gebundene Variablen
  - SQL
    - Anmerkung zu Unteranfragen
    - Gruppierung / Aggregation
    - All-Quantifizierung
- Relationale Entwurfstheorie
  - ...

Anmerkungen zur relationalen Algebra

- Geklammerter Ausdruck vs. Operatorbaum
- Umbenennung







Anmerkung: Natural-Join vs. allgemeiner Theta-Join

	Natural	Theta
Inner	$\bowtie$	$\bowtie_\theta$
Outer	$\mathbin{⟗}$, $\mathbin{⟕}$, $\mathbin{⟖}$	$\mathbin{⟗_\theta}$, $\mathbin{⟕_\theta}$, $\mathbin{⟖_\theta}$
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.

Kalkül: Freie vs. gebundene Variablen (1/2)

$R : \{[a, b, c]\}, S : \{[d, e, f]\}$

Relationale Algebra: $Q_1 := R \bowtie_{a=d} S$

r und s sind gebunden → keine Quantoren

Tupelkalkül: $Q_1 := \{ r \circ s \mid r \in R \wedge s \in S \wedge r.a = s.d \}$

Projektion

Bedingungen, die gelten müssen

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

Kalkül: Freie vs. gebundene Variablen (2/2)

$R : \{[a, b, c]\}$, $S : \{[d, e, f]\}$

Relationale Algebra: $Q_2 := R \ltimes_{a=d} S$

Tupelkalkül: $Q_2 := \{ r \mid r \in R \ \wedge \exists s \in S(r.a = s.d) \}$

„für das gilt"

„Gültigkeitsbereich" von s

Domänenkalkül: $Q_2 := \{ [a, b, c] \mid [a, b, c] \in R \ \wedge \exists e, f([a, e, f] \in S) \}$

„Gültigkeitsbereich" von e und f

Bitte beachten:

Alle Domänen- / Tupelvariablen tauchen entweder in der Projektion auf, oder wurden explizit quantifiziert.

Anmerkung zu Unteranfragen in SQL

```
select * from R
where R.x = (select y from S where ... )
```

Die Unteranfrage muss genau einen skalaren Wert ausgeben.

```
select * from R
where R.x in (select y from S where ... )
```

Die Unteranfrage muss eine Menge von skalaren Werten ausgeben.

Übung: Gruppierung/Aggregation in SQL

"Gute Studenten": Finde Studenten in höheren Semestern (sem > 5) mit einem Notenschnitt < 2.5. Geben Sie **MatrNr**, **Name** und **Notenschnitt** aus.

```
--"Gute Studenten":
-- Finde Studenten in höheren Semestern (sem > 5) mit einem
-- Notenschnitt < 2.5. Geben Sie MatrNr, Name und Notenschnitt aus.
select s.matrnr, s.name, avg(p.note*1.0)
  from studenten s, pruefen p
 where s.matrnr = p.matrnr
   and s.semester > 5
group by s.matrnr, s.name
having avg(p.note*1.0) < 2.5
```

Übung: All-Quantifizierung in SQL

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

$A \Rightarrow B \equiv \neg A \vee B$

Finde Studenten, die ALLE Vorlesungen gehört haben.

Tupelkalkül: ! kein ALL in SQL lediglich EXISTS und NOT, auch keine $\Rightarrow$, nur AND und OR

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

$\{s \mid s \in S \wedge \neg\exists v \in V(\neg\exists h \in hören(\ldots))\}$

SQL:

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

Relationale Entwurftheorie

„bestimmt funktional" oder „bestimmt eindeutig"

Bedingungen aus der realen Welt; Bsp.: MatrNr → Name

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

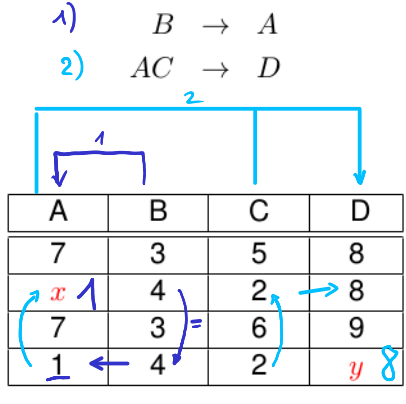
Übung: Relationenausprägung vervollständigen

FD - Sudoku

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

1) $B \rightarrow A$

2) $AC \rightarrow D$



Funktionale Abhängigkeiten

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

Axiome von Armstrong:

triviale FDs

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

i.d.R. werden nicht alle geltenden FDs angegeben; insbesondere nicht die trivialen

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: $\alpha' \to \mathcal{R}$

kein Attribut aus κ kann entfernt werden, ohne dass die Schlüsseleigenschaft verloren geht.

- I.A. existieren mehrere Super- und Kandidatenschlüssel.
- Man muss sich bei der Realisierung für einen Kandidatenschlüssel entscheiden, dieser wird dann **Primärschlüssel** genannt.

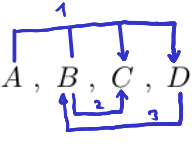
in SQL

- 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 (i.A. zu aufwendig!)
- Besser: Die **Attributhülle** $AH(\alpha)$ bestimmen.

- Beispiel: $\mathcal{R} = \{ A , B , C , D \}$, mit $F_{\mathcal{R}} = \{\underbrace{AB \to CD}_{1}, \underbrace{B \to C}_{2}, \underbrace{D \to B}_{3}\}$



$AH(\{D\})$: $D \overset{3}{\rightsquigarrow} B,D \overset{2}{\rightsquigarrow} \underbrace{B,C,D}_{\neq \mathcal{R}}$ ✗ kein Schlüssel in $\mathcal{R}$

$AH(\{A,D\})$: $AD \overset{3}{\rightsquigarrow} ABD \overset{1}{\rightsquigarrow} \underbrace{ABCD}_{=\mathcal{R}}$ ✓ Schlüssel in $\mathcal{R}$

$AH(\{A,B,D\})$: ✓ Schlüssel, da die Teilmenge $\{A,D\}$ bereits ein Schlüssel ist.
↳ jedoch nicht minimal => kein Kandidatenschlüssel

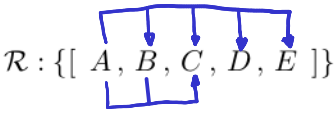
Normalformen: 1NF ⊃ 2NF ⊃ 3NF ⊃ BCNF ⊃ 4NF

Verletzung der 1. NF

z.B. $R = \{a, \{b, \ldots b_n\}, c\}$

- **1. NF**: Attribute haben nur atomare Werte, sind also nicht mengenwertig.
- **2. NF**: Jedes Nichtschlüsselattribut (NSA) ist voll funktional abhängig von jedem Kandidatenschlüssel.
  (Attr., die in keinem Kandidatenschlüssel enthalten sind).
  Bsp: $AB \to C$, $B \to C$ ↯ also $B \not\to C$ Verletzung der 2. NF
  - β hängt **voll funktional** von α ab ($\alpha \to \beta$), gdw. $\alpha \to \beta$ und es existiert kein $\alpha' \subset \alpha$, so dass $\alpha' \to \beta$ gilt.
- **3. NF**: Frei von transitiven Abhängigkeiten (*in denen NSAe über andere NSAe vom Schlüssel abhängen*).
  Schlüssel → NSA_1 → NSA_2 ↯
  - 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.

Übung: Höchste NF bestimmen



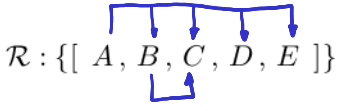
$$A \rightarrow BCDE$$
$$AB \rightarrow C$$

Kandidatenschlüssel müssen bestimmt werden

1. NF: ✓
2. NF: ✓ $\mathcal{X} = \{A\}$ NSAe $= \{B, C, D, E\}$
3. NF: ✓ keine FD verletzt die 3. NF
BCNF: ✓

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

Übung: Höchste NF bestimmen (2)



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

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

1.NF: ✓

2.NF: ✓ $K = \{A\}$ $NSAe = \{B, C, D, E\}$

∵ da nur ein K mit nur einem Attribut

3.NF: ✗ $B \rightarrow C$ nicht in K

↑ kein Superschlüssel

FD verletzt 3.NF

Schema in 3. NF überführen

Fortsetzung in der nächsten Zü.

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, wo alle Schemata in 3.NF sind.
  - Die Zerlegung ist **abhängigkeitsbewahrend** und **verlustfrei**.

Übung: Synthesealgorithmus (1)

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

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

Übung: Synthesealgorithmus (2)

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

Schema in BCNF überführen

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 $Z := (Z - \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i.1}\} \cup \{\mathcal{R}_{i.2}\}$

BCNF-Dekompositionsalgorithmus

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