Grundlagen: Datenbanken

Zentralübung / Wiederholung / Fragestunde

Christoph Anneser Moritz Sichert Lukas Vogel

gdb@in.tum.de

WiSe 2019 / 2020

Diese Folien finden Sie online.

Die Mitschrift stellen wir im Anschluss online.

Agenda

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

Hinweise zur Klausur

Termine

- 1. Klausurtermin - **Fr. 14.02.2020, 16:00 bis 17:30 Uhr**
- Notenbekanntgabe - **vsl. Di. 03.03.2020 (ab Mittag)**
- Einsicht - **vsl. 04.-06.03.2020**
- Anmeldung zur 2. Klausur von - **18.03.2020**, bis **30.03.2020**
- 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 $\mapsto$ Sitzplatz, KEINE Namensnennung)
- Einsichtnahme (Instruktionen in Moodle, nach Notenbekanntgabe)
- Bonus: Gilt für beide Klausuren.

Stoffübersicht (1)

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

Datenbankentwurf / ER-Modellierung

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

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, 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 1, 5, 6
  - TID-Konzept
  - Indexstrukturen
    - B-Baum, B+-Baum
    - Erweiterbares Hashing

- **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

Stoffausblick verbleibende Wochen

- **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/¬steal, Einbringstrategie force/¬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

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 ($\wedge, \vee, \neg, \Rightarrow$) $a \Rightarrow b \quad \Leftrightarrow \quad \neg a \vee b$
- 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.

$$\{ s \mid s \in \text{Studenten} \wedge \neg \exists h \in \text{hören} \; (s.\text{matrnr} = h.\text{matrnr}) \}$$

$$\forall h \in \text{hören} \; \neg (s.\text{matrnr} = h.\text{matrnr})$$

$$\{ [m, n, s] \mid [m, n, s] \in \text{Studenten} \wedge \neg \exists hm, v \; ([hm, v] \in \text{hören} \wedge (m = hm)) \}$$

$\neg \exists \Leftrightarrow \forall$

$\neg \exists x : p(x) \quad \Leftrightarrow \quad \forall x : \neg p(x)$

$\neg \forall x : p(x) \quad \Leftrightarrow \quad \exists x : \neg p(x)$

$\forall h \in \text{hören} \; (h.\text{matrnr} \neq s.\text{matrnr})$

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

Übung: Tupel- / Domänenkalkül (2)

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

(123, GDB, 3)

$$\{ s \mid s \in \text{Studenten} \wedge \forall v \in \text{Vorlesungen} (v.\text{SWS} > 4 \Rightarrow \exists h \in \text{hören} (h.\text{matrnr} = s.\text{matrnr} \wedge h.\text{vorlnr} = v.\text{vorlnr})) \}$$

alle Stud. die GDB hören:

$$\{ [m, n, s] \mid [m, n, s] \in \text{Studenten} \wedge \exists v ([m, v] \in \text{hören}) \}$$

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

$\{A, B\} \rightarrow \{C, D\}$

$AB \rightarrow CD$

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	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 $\alpha \to \beta$
- 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 $\alpha \to \gamma$

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 α zudem noch minimal, ist es auch ein **Kandidatenschlüssel** (meist mit κ 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.

R: {A,B,C} ABC → ABC

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.

$AH(\alpha) \rightarrow R$

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

$\emptyset \rightarrow F$

$\alpha : \{AD, AB\}$

$AH(\{D\})$: DBC

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

$AD \rightarrow R$

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

$ABD \rightarrow R$

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

$AH(\{D\}) : DBC$

$AH(\emptyset) : \emptyset$

Mehrwertige Abhängigkeiten

multivalued dependencies (MVDs)

$AC \twoheadrightarrow BD$

Q.: $\{A, B, C, D\}$

$\alpha = AC \quad \beta = B \quad \gamma = D$

$AC \twoheadrightarrow B$

"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 $\gamma = \emptyset$)

Beispiel: Mehrwertige Abhängigkeiten

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

ProfessorIn →→ Vorlesung | ProfessorIn →→ Assistent

ProfessorIn	Vorlesung	AssistentIn
K	GDB	Linnea
K	WebDB	Linnea
K	GDB	Lukas
K	WebDB	Lukas
K	ERDB	Linnea
K	ERDB	Lukas

K, GDB, Linnea? ✓
K, WebDB, Lukas?
K, ERDB, Lukas?
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.
  - 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**: 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

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

$$A \rightarrow BCDE$$

$$AB \rightarrow C$$

- ☑ 1. NF
- ☑ 2. NF
- ☑ 3. NF
- ☑ BCNF
- ☐ 4. NF
- ☐ keine der angegebenen

Übung: Höchste NF bestimmen (2)

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

$k : \{A\}$

$A \rightarrow B$

$\vdots$

$A \rightarrow E$

$A \rightarrow BCDE$

$B \rightarrow C$

- ☑ 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

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

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

$R_1 : \{A, B, E, F\}$

$R_2 : \{B, C, E, F\}$

$R_3 : \{A, D\}$

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

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

$F_{\mathcal{R}} = \{B \to AD, DEF \to B, C \to AE\}$

$k = \{CDF, CBF, \ldots$

$R_1 = \{A, B, D\}$ ✓

$R_2 = \{B, C, E, F\}$ $(C \to E)$

$R_{2.1} = \{C, E\}$ ✓

$R_{2.2} = \{B, C, F\}$ ✓

$Z = \{R_1, R_{2.1}, R_{2.2}\}$

Erweiterbares Hashing

Hashfunktion ***h*: *S*** → ***B***

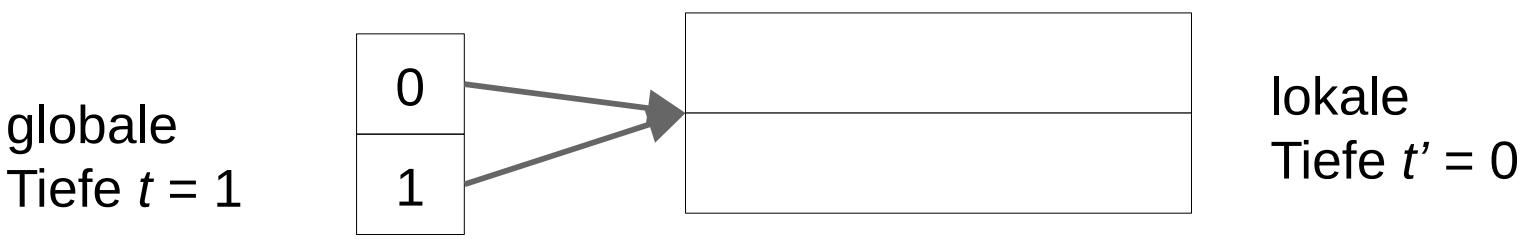
Schlüssel — Bucket

wir betrachten die **Binärdarstellung** des Hashwerts

$h(x) = dp$

unbenutzte Bits

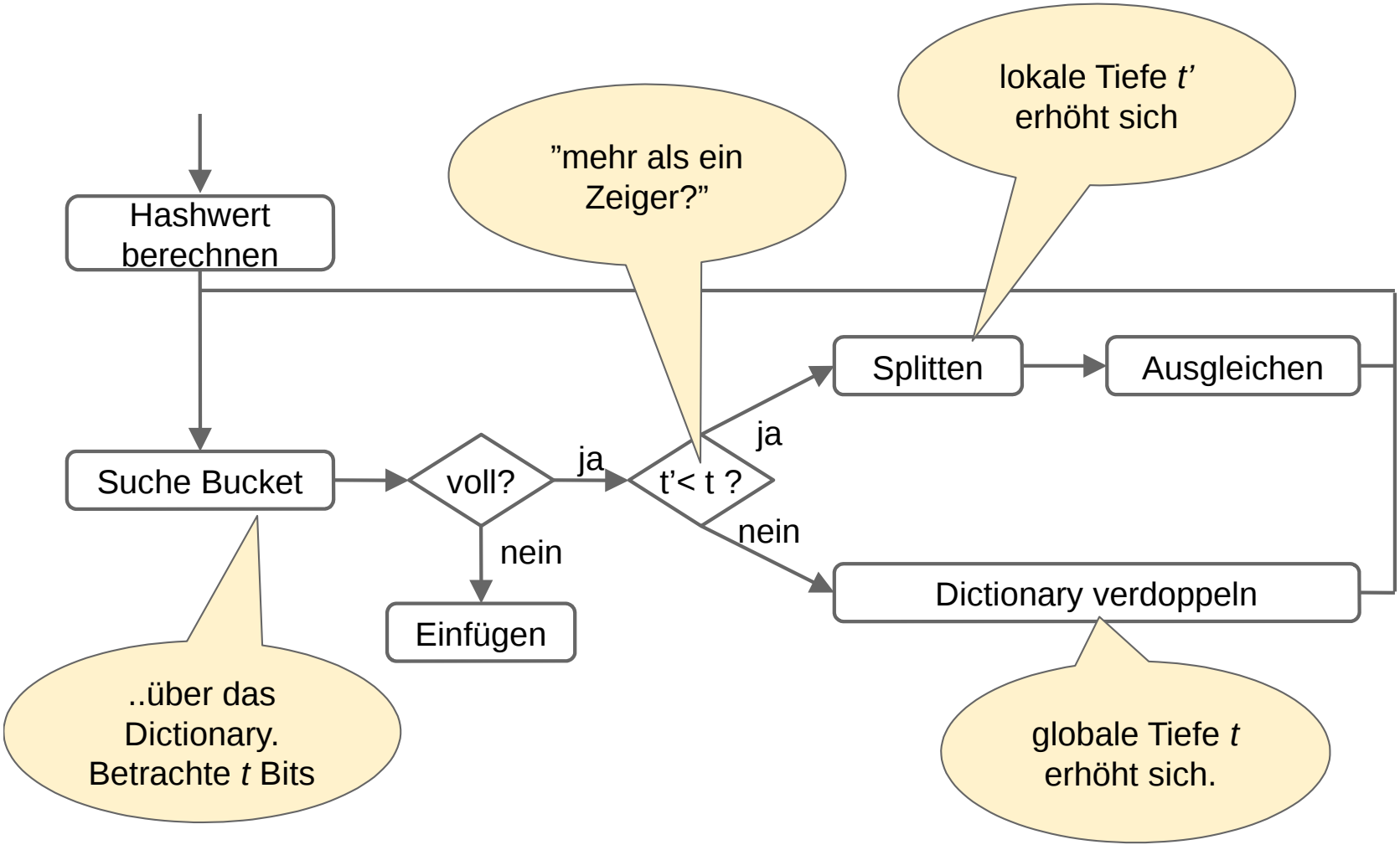
Anzahl betrachteter Bits
= globale Tiefe des Dictionaries



Dictionary
(Verzeichnis)

Bucket (Behälter)
Platz für *n* Einträge
hier *n*=2

Erweiterbares Hashing / Einfügen



Übung: Erweiterbares Hashing / Einfügen

x	*h(x)*
A	1100
B	0100
C	1101
D	1010

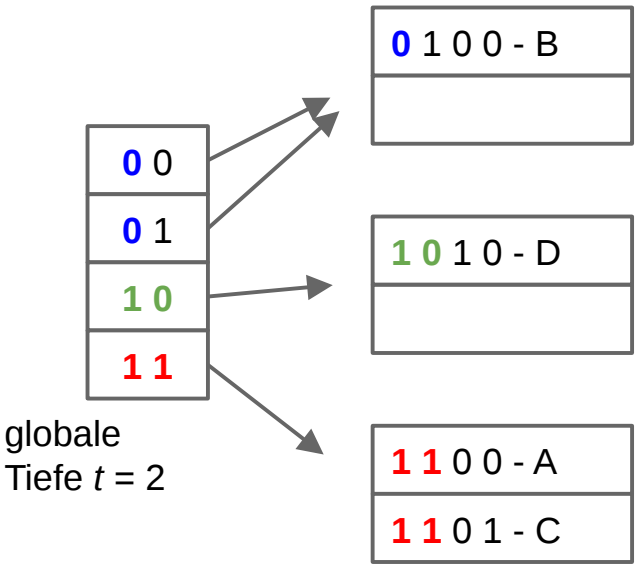
Übung: Erweiterbares Hashing / Einfügen

x	$h(x)$
A	1100
B	0100
C	1101
D	1010

Erweiterbares Hashing / Lösung

x	*h(x)*
A	1100
B	0100
C	1101
D	1010

d *p*



lokale
Tiefe *t'* = 1

t' = 2

t' = 2

in einem Bucket mit Tiefe *t'*, stimmen (mindestens) die *t'* führenden Bits der Hashwerte überein

Anfrageoptimierung

- Kanonische Übersetzung:
  - Projektion für alle Attribute in `select`
  - Selektion für `where`-Bedingung
  - Kreuzprodukte für alle Relationen in `from`
- Logische Optimierung, Ziel: Verringern der Zwischenergebnisse, Verwendung der Äquivalenzregeln:
  - Frühe Selektion („push down“)
  - Selektion + Kreuzprodukt ersetzen durch Join
  - Joinreihenfolge optimieren
- Physische Optimierung: Wahl des Joinalgorithmus:
  - Nested-Loop-Join
  - Sort-Merge-Join
  - Hash-Join
  - Index-Join
- Iteratormodell
  - `open()`, `next()`, `close()`

Übung: Anfrageoptimierung

Geben Sie die kanonische Übersetzung der folgenden SQL-Anfrage an und optimieren Sie diese logisch:

```
select distinct s.name
from studenten s, hören h, vorlesungen v
where
    s.matrnr = h.matrnr and
    h.vorlnr = v.vorlnr and
    v.titel = 'Grundzüge'
```

Übung: Anfrageoptimierung (2)

Angenommen

- $|s| = 10000$
- $|h| = 20 \cdot |s| = 200000$
- $|v| = 1000$
- 10% der Studenten haben 'Grundzüge' gehört

Dann ergeben sich

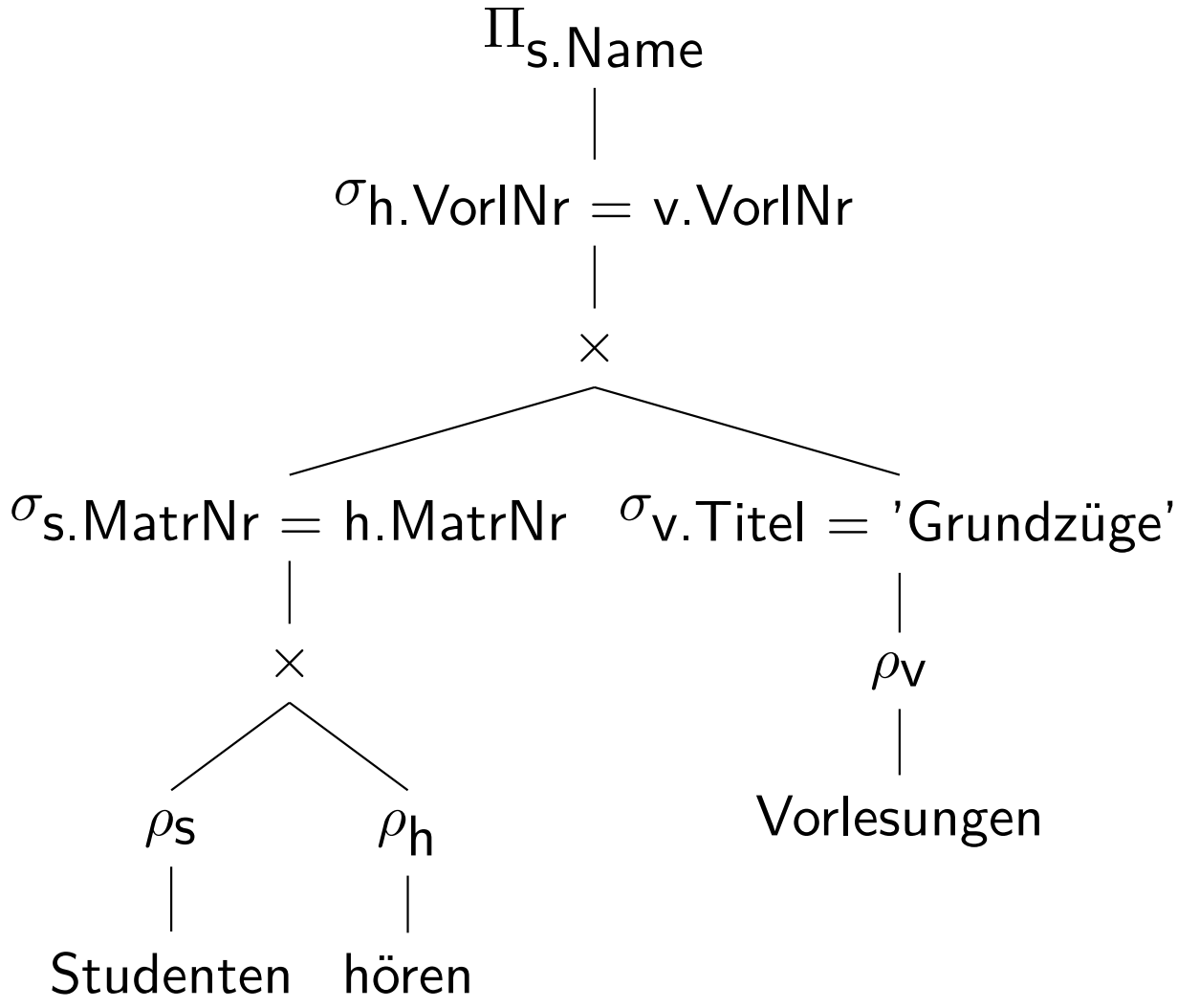
- $|s \times h \times v| = 10000 \cdot 20 \cdot 10000 \cdot 1000 = 2 \cdot 10^{12}$

Nach der Selektion verbleiben noch

- $|\sigma_p(s \times h \times v)| = 0,1 \cdot |s| = 1000$

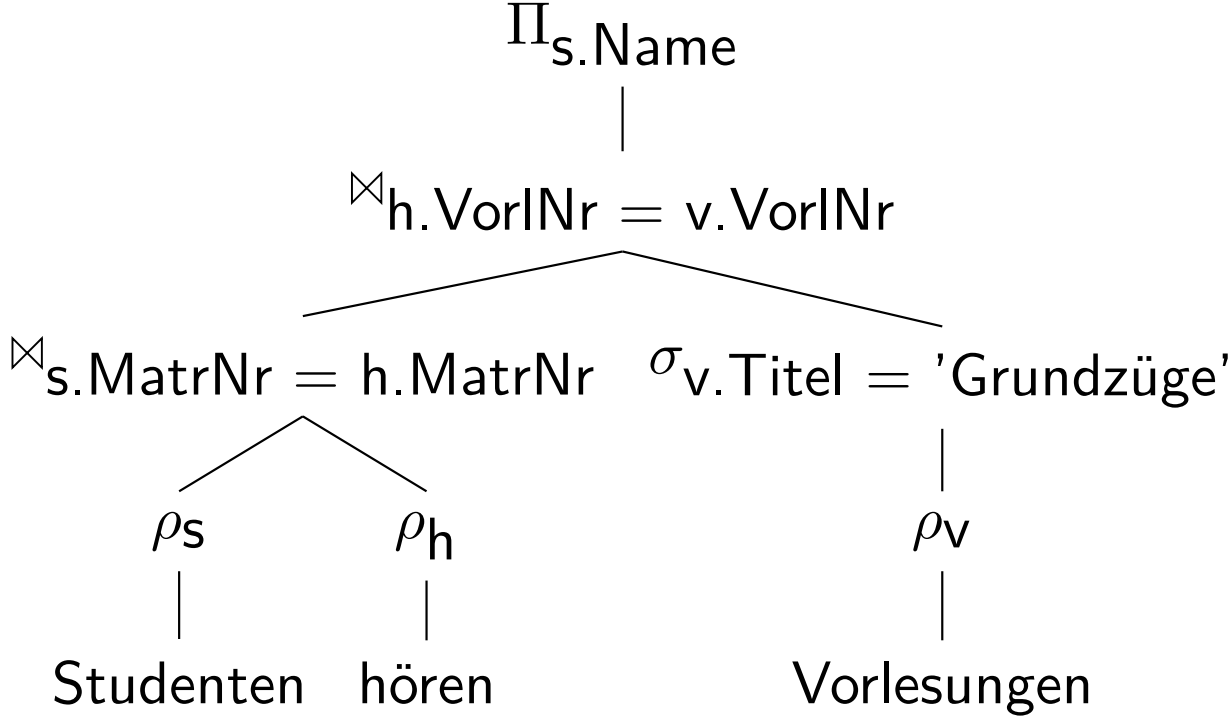
Übung: Anfrageoptimierung (3)

Optimierung 1: Selektionen frühzeitig ausführen (*push selections*):



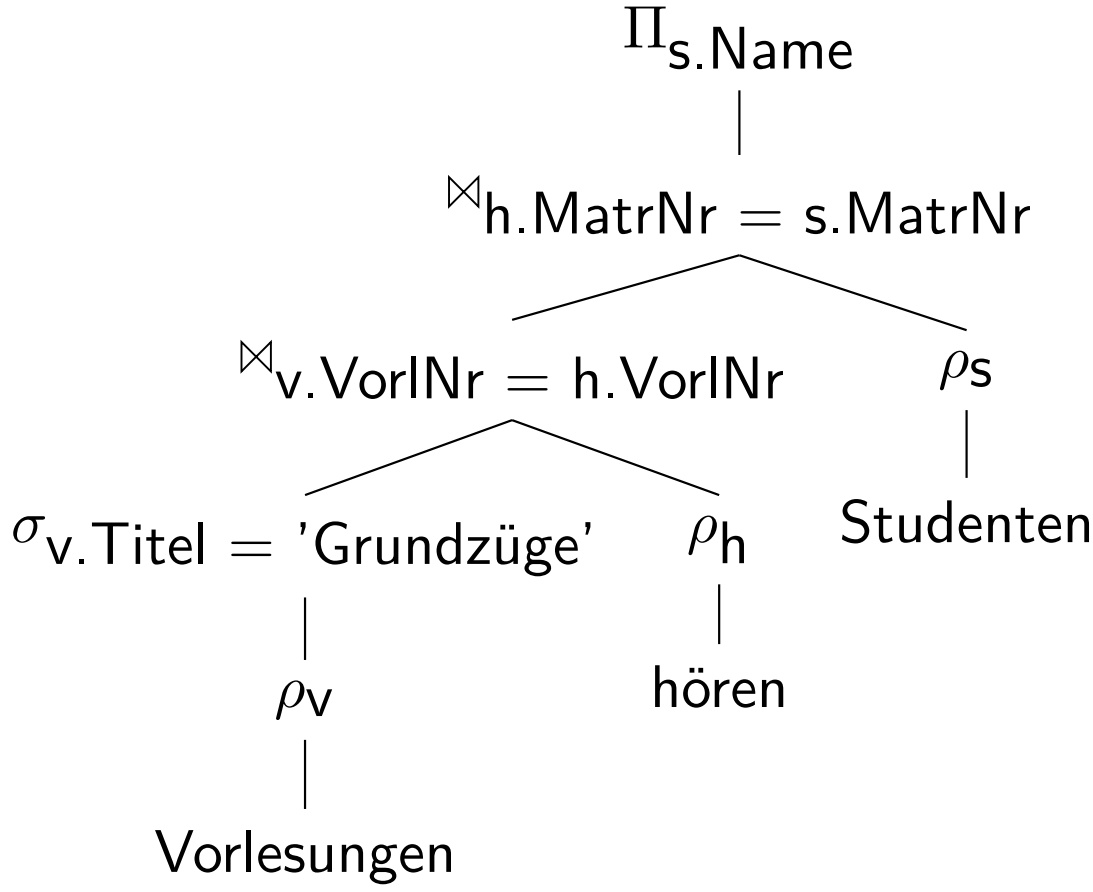
Übung: Anfrageoptimierung (4)

Optimierung 2: Kreuzprodukte durch Joins ersetzen (*introduce joins*):



Übung: Anfrageoptimierung (5)

Optimierung 3: Joinreihenfolge optimieren (*join order optimization*), so dass die Zwischenergebnismengen möglichst klein sind:



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 bv as (
    select vor.vorgaenger as VorlNr
    from Vorlesungen v, Voraussetzen vor
    where v.Titel = 'Bioethik' and
          v.VorlNr = vor.nachfolger
  union all
    select vor.vorgaenger as VorlNr
    from bv, Voraussetzen vor
    where bv.vorlnr = vor.nachfolger) select ... from ...
```

base-case

Rekursionsschritt

```
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'
```

Wir wünschen viel Erfolg. :-)