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 $\mapsto$ 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
  - 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

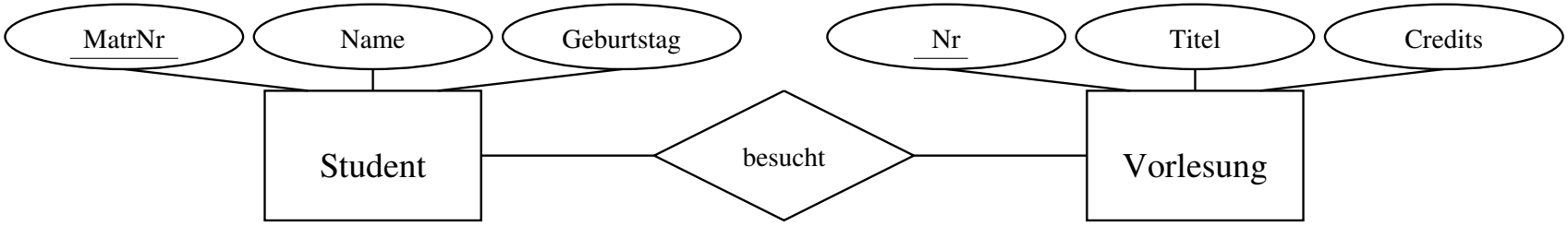
Stoffübersicht (5)

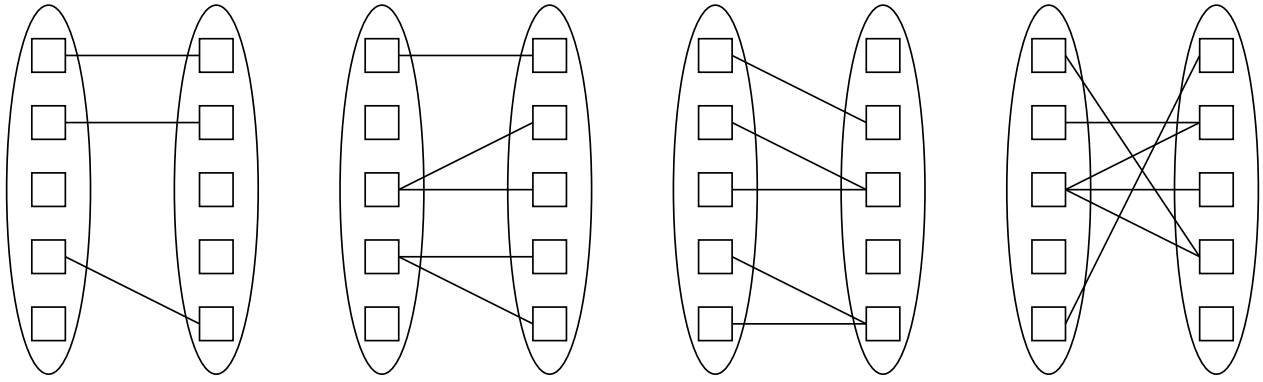
- **Transaktionsverwaltung**
  - BOT, read, write, commit, abort
  - Rollback (R1 Recovery)
  - ACID-Eigenschaften
- **Fehlerbehandlung (Recovery)**
  - Fehlerklassifikation (R1 - R4)
  - Protokollierung: Redo/Undo, physisch/logisch, Before/After-Image, WAL, LSN
  - Pufferverwaltung: Seite, FIX, Ersetzungsstrategie 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
- 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

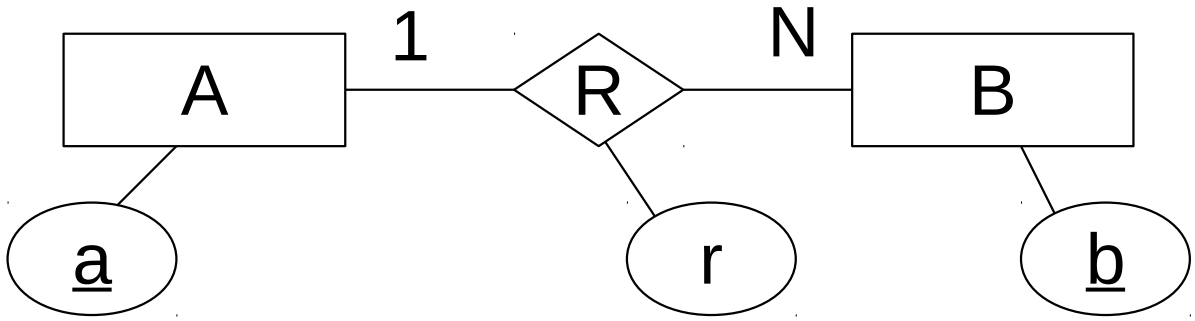
Datenbankentwurf



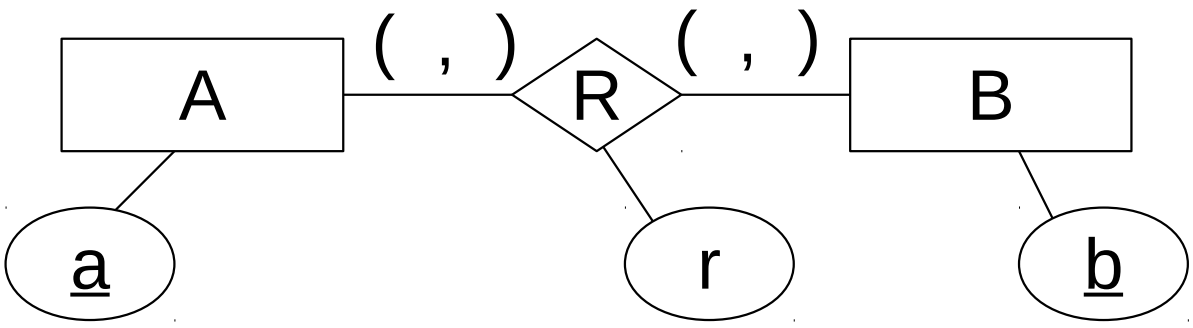


1:1 1:N N:1 M:N

ER-Modell in Schema überführen und verfeinern



(Min,Max) - Angaben



R		
a	b	r

Relationale Algebra

Algebraische Operatoren:

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

- Schreibweise Tupelkalkül: $\{\, t \mid p(t) \,\}$ bzw. $\{\, [t.a1, t.a2] \mid p(t) \,\}$
- Schreibweise Domänenkalkül: $\{\, [a1, a2, a3] \mid p(a1, a2, a3) \,\}$
- 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.

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 (DDL I)

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

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.

Übung: SQL (Aggregation)

Wieviele Studierende gibt es pro Semester?

Übung: SQL (Aggregation / Allquantor)

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

Übung: SQL (Rekursion I)

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

Übung: SQL (Rekursion II)

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

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).

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.

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

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

Funktionale Abhängigkeiten

Seien 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. $\alpha \rightarrow \mathcal{R}$
- 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.

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

$AH(\{D\})$:

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

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

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 $\gamma = \emptyset$)

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 (NSA) ist voll funktional abhängig von jedem Kandidatenschlüssel.
  - β hängt **voll funktional** von α ab ($\alpha \overset{\bullet}{\rightarrow} \beta$), 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
  - 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

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

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

- ○ 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

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

Erweiterbares Hashing

Hashfunktion ***h*: *S*** → ***B***

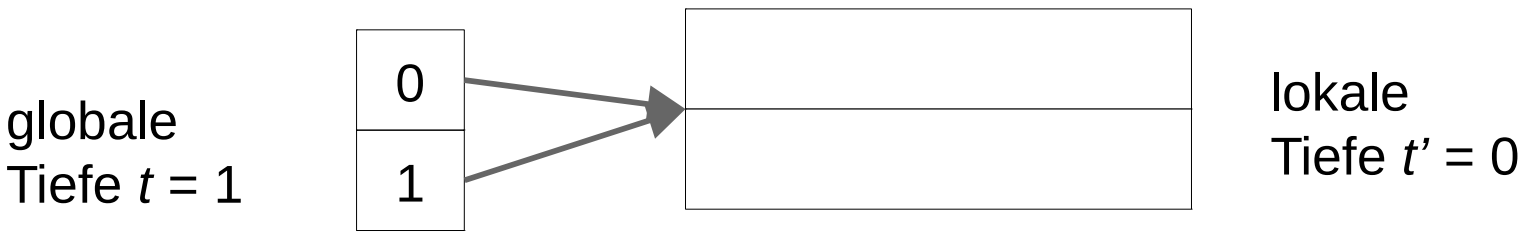
Schlüssel — Bucket

wir betrachten die **Binärdarstellung** des Hashwerts

$h(x) = dp$

unbenutzte Bits

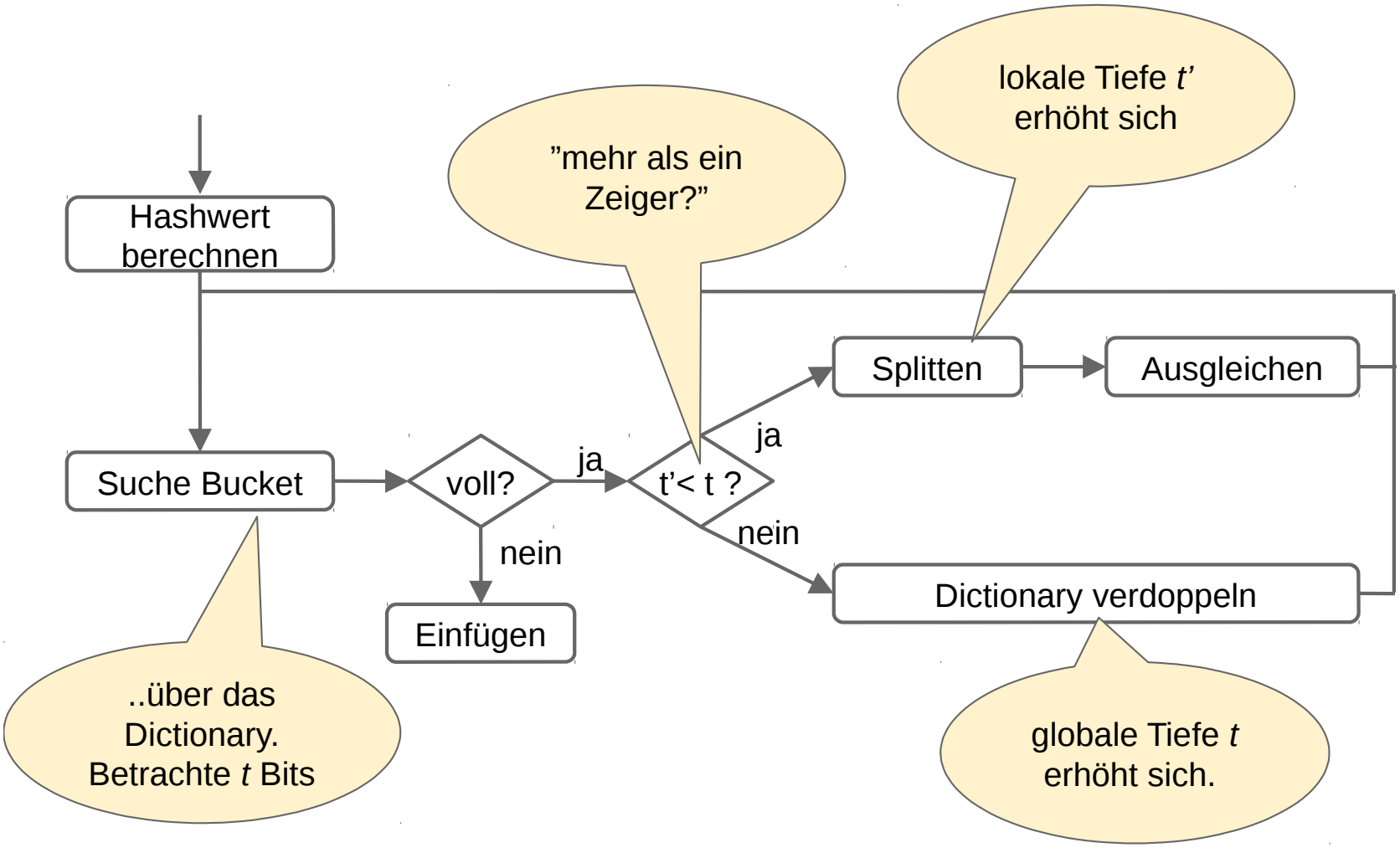
Anzahl betrachteter Bits
= globale Tiefe des Dictionaries



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

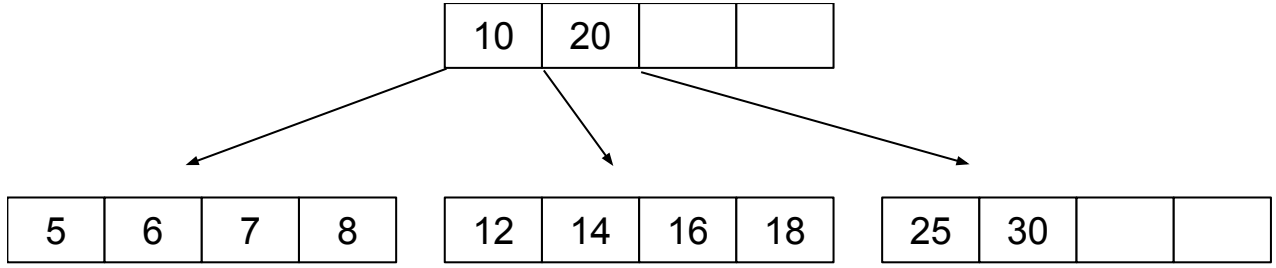
x	*h(x)*
A	1100
B	0100
C	1101
D	1010

B-Baum

- Für Hintergrundspeicher konzipiert (1 Knoten $=$ 1 Seite, Knotenkapazität $>$ 1000)
- Hat Grad k: Jeder Knoten (außer Wurzel) mind. k, maximal $2k$ Elemente
  - Bei Überfüllung nach Einfügen: Aufteilen in 2 Knoten.
  - Bei Unterbelegung nach Löschen: Ausgleich mit Nachbarn, oder Verschmelzung wenn Nachbar minimal belegt.
  - Aufteilen oder Verschmelzen setzt sich rekursiv bis zur Wurzel fort.
- B+-Bäume sind *hohl*, haben Werte nur in den Blättern:
  - hat daher Grad (k, k^*). k für innere Knoten, k^* für Blattknoten.

Übung: B-Baum (1)

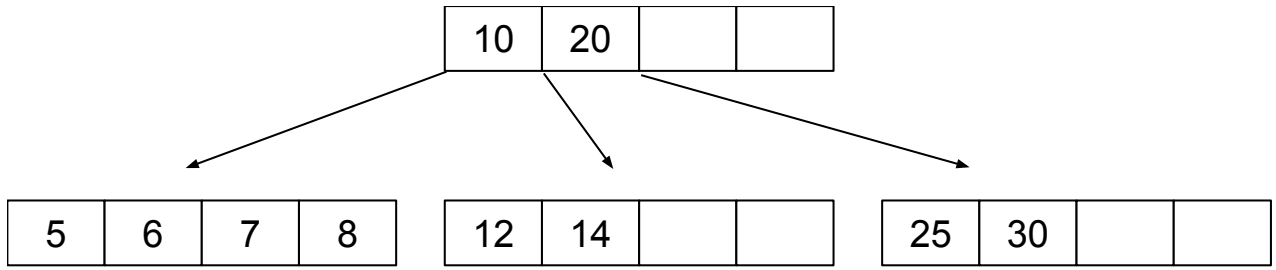
Fügen Sie die Zahl **9** in den folgenden B-Baum ein.



Zeichnen Sie den vollständigen, resultierenden Baum inklusive aller Referenzen.

Übung: B-Baum (2)

Löschen Sie die Zahl **30** im folgenden B-Baum.



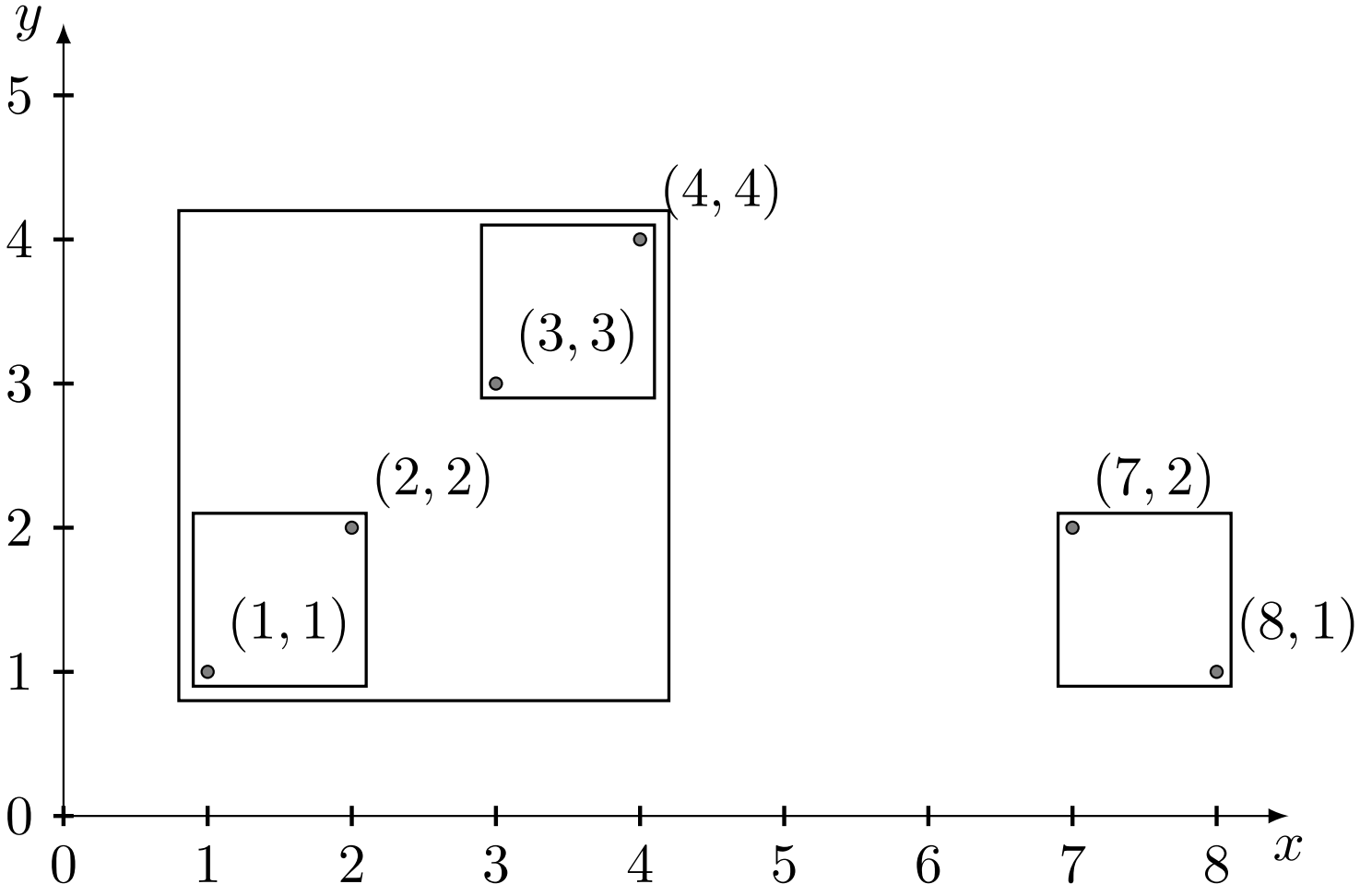
Zeichnen Sie den vollständigen, resultierenden Baum inklusive aller Referenzen.

R-Baum

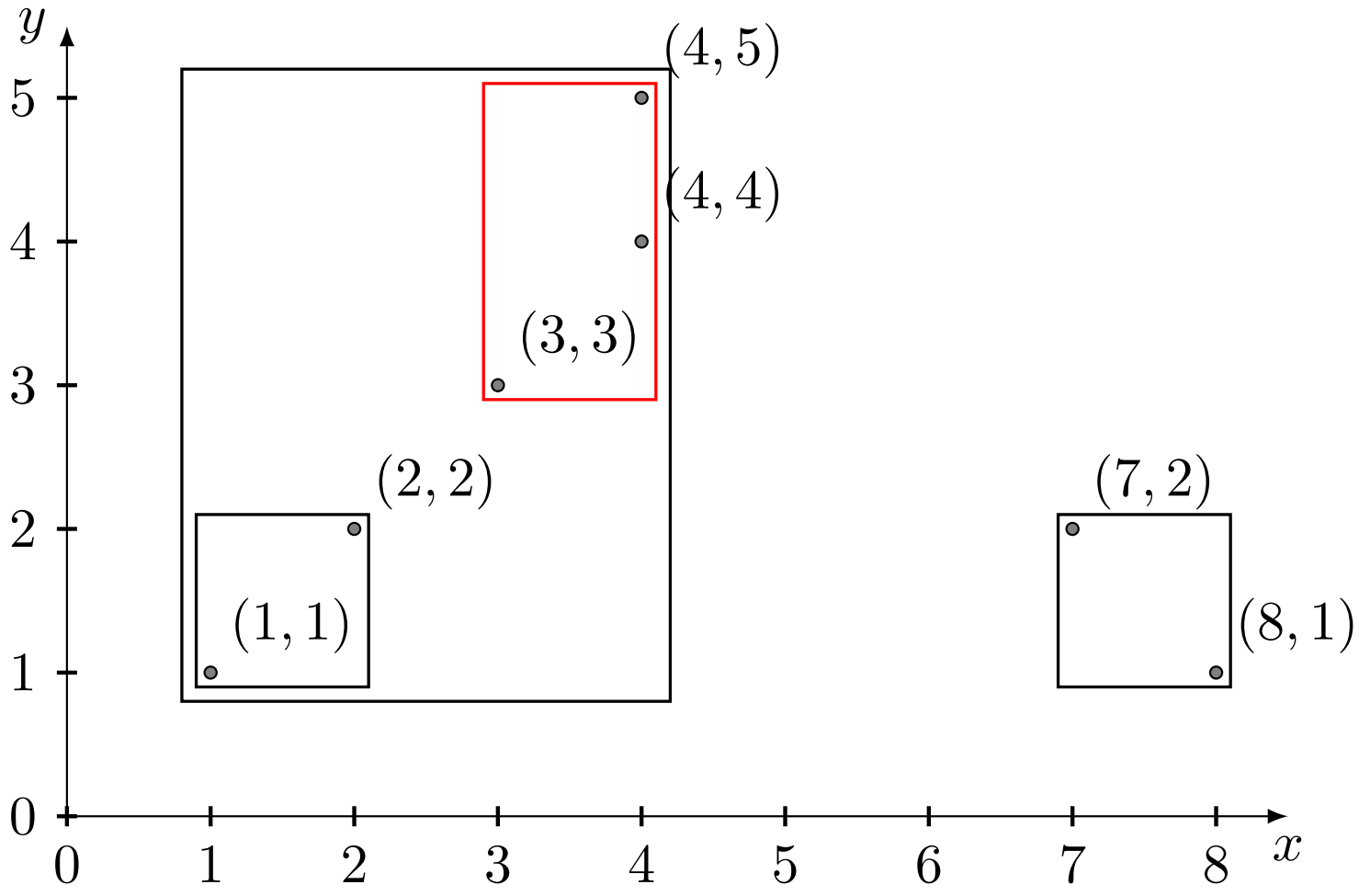
- Mehrdimensionale Indexstruktur: Punkte in n-dimensionalem Datenraum werden in Knoten aufgeteilt
- Funktionsweise ähnlich zu B+-Baum
- Split nicht eindeutig, da es nicht „die beste“ Möglichkeit gibt, Boxen aufzuteilen
- Punkt- und Bereichsabfragen müssen meist mehrere Pfade traversieren, da sich Boxen überschneiden können

Übung: R-Baum

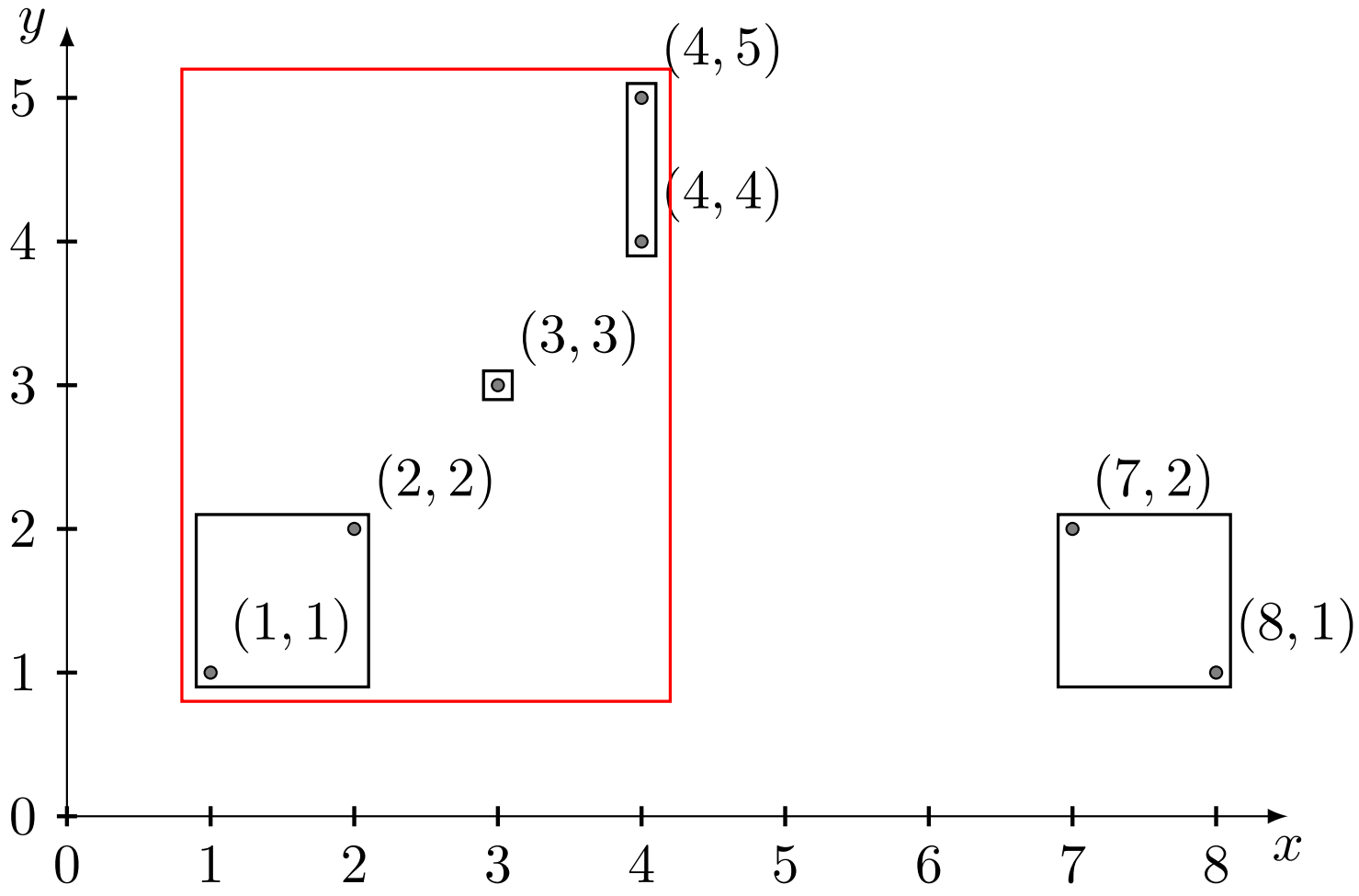
Fügen Sie den Wert $(4, 5)$ in den untenstehenden R-Baum mit der Knotenkapazität 2 ein und zeichnen Sie den resultierenden Datenraum.



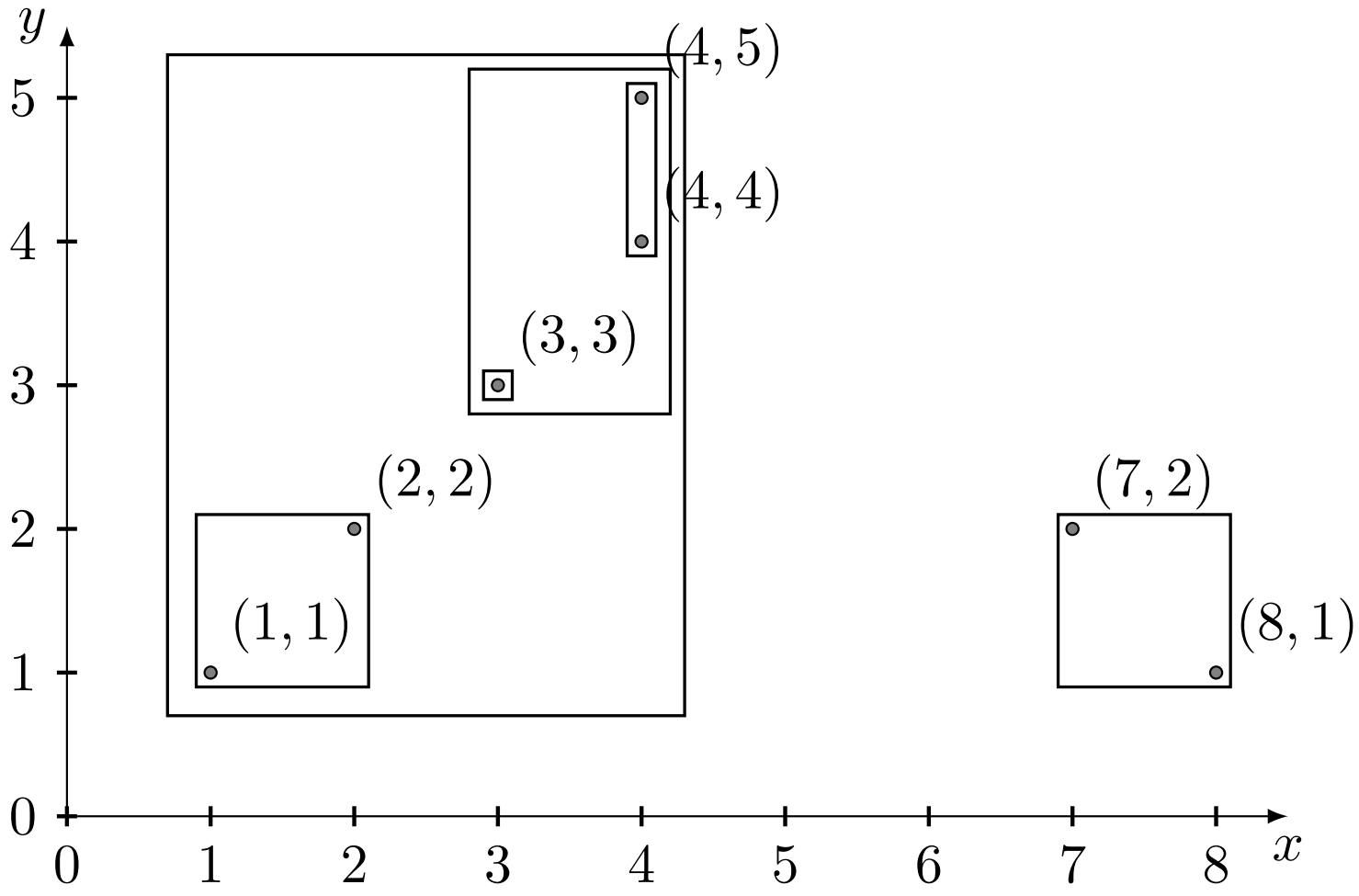
Übung: R-Baum



Übung: R-Baum



Übung: R-Baum



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

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

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

Dann ergeben sich

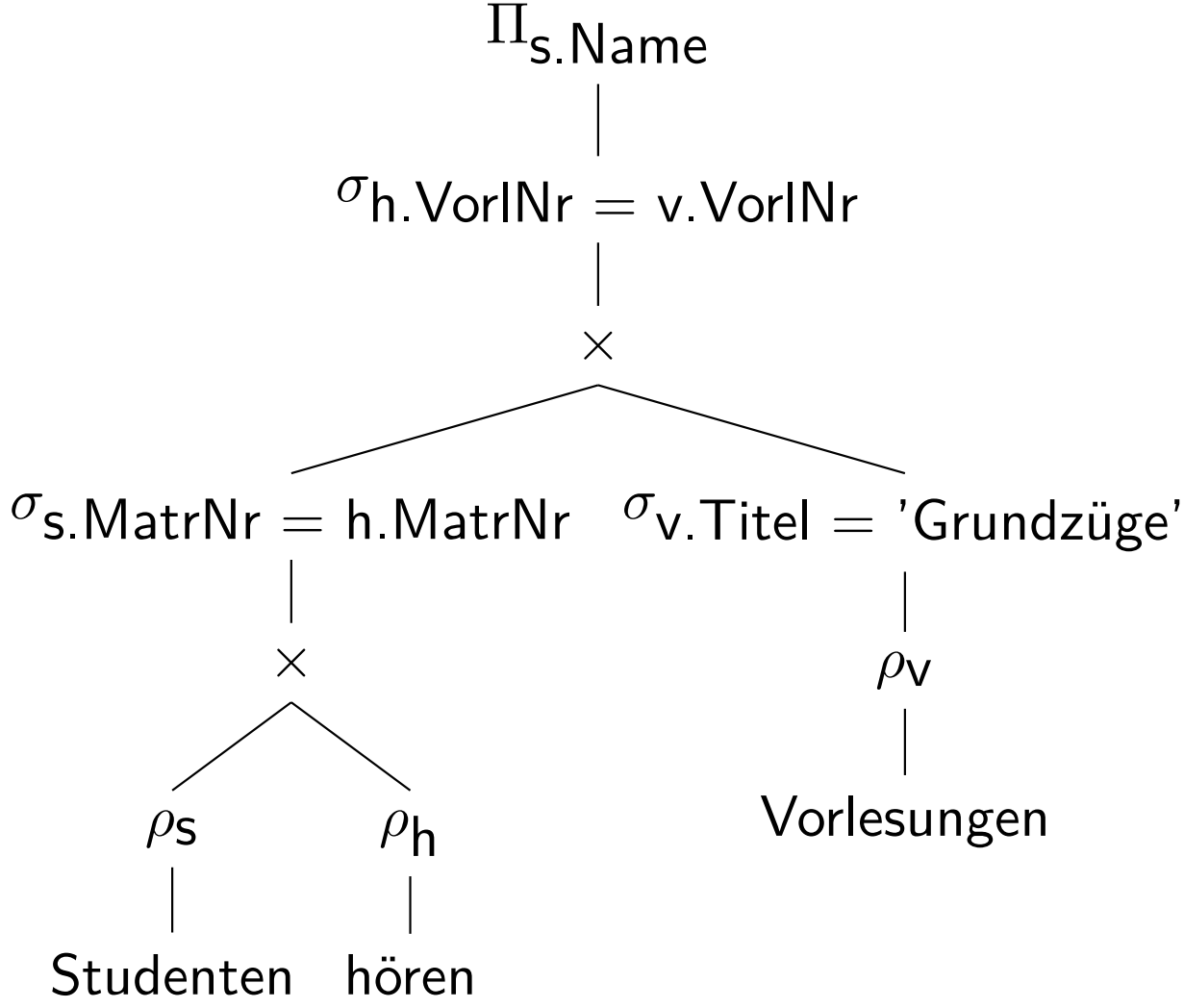
- $|s \times h \times v| = 10000 \cdot 20 \cdot 10000 \cdot 1000 = 2 \cdot 10^{12}$

Nach der Selektion verbleiben noch

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

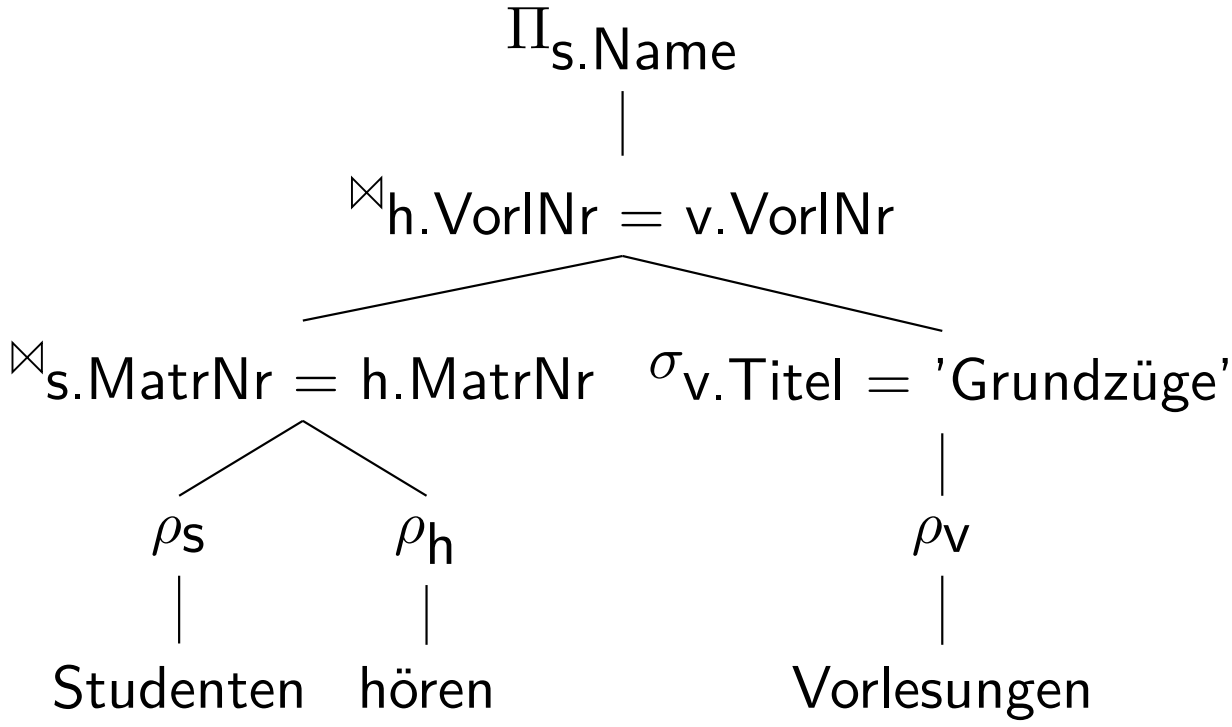
Übung: Anfrageoptimierung (3)

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



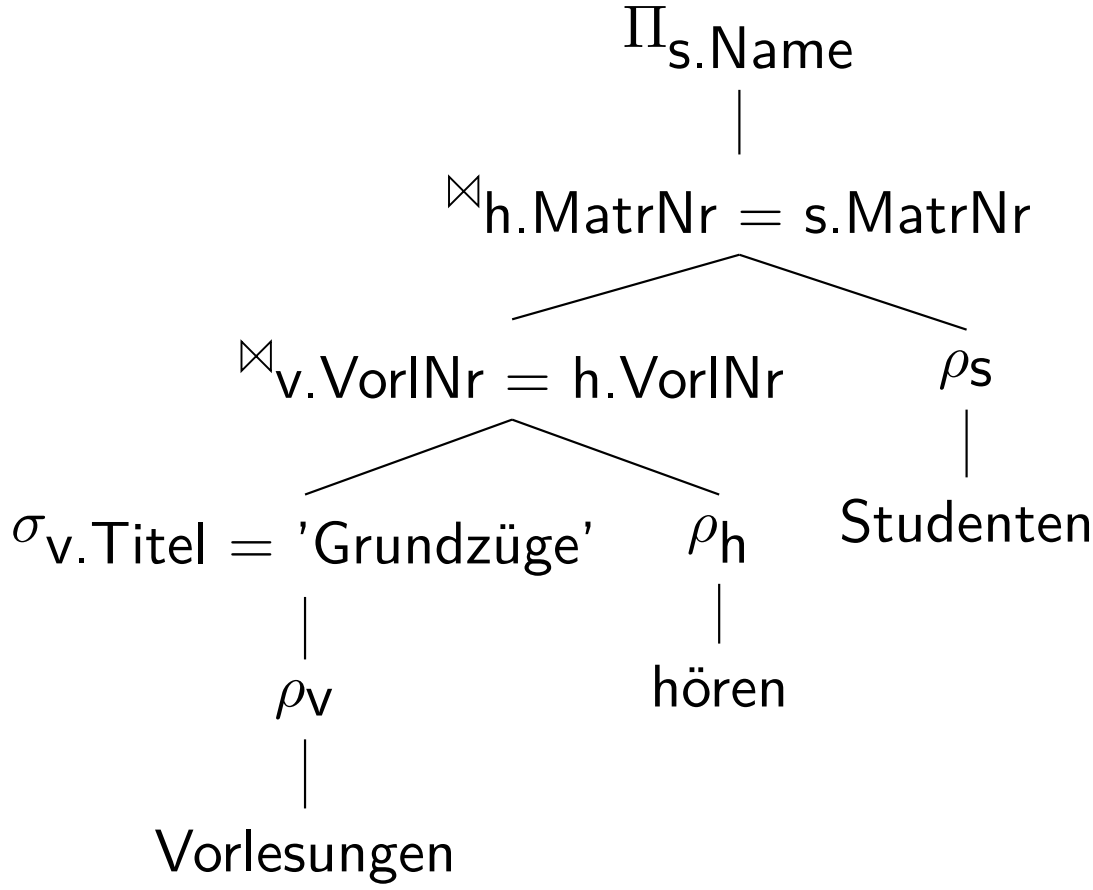
Übung: Anfrageoptimierung (4)

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:



Fragen?

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

Wir wünschen viel Erfolg. :-)