TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS19/20

Christoph Anneser, Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws1920/grundlagen/

Blatt Nr. 12

Da diese Woche keine neuen Themen in der Vorlesung behandelt wurden, enthält dieses Blatt Aufgaben, die in der Zentralübung häufig gewünscht wurden.

Hausaufgabe 1

„Fleißige Studenten“: Formulieren Sie eine SQL-Anfrage, um die Studenten zu ermitteln, die mehr SWS belegt haben als der Durchschnitt. Berücksichtigen Sie dabei auch Totalverweigerer, die gar keine Vorlesungen hören.

Lösung:

Folgende SQL-Anfrage ermittelt die fleißigen Studenten:

```
select s.*
from Studenten s
where s.MatrNr in
  (select h.MatrNr
   from hoeren h join Vorlesungen v on h.VorlNr = v.VorlNr
   group by h.MatrNr
   having sum(SWS) >
      (select sum(cast(SWS as decimal(5,2)))/count(
         distinct(s2.MatrNr))
       from Studenten s2
          left outer join hoeren h2 on h2.MatrNr = s2.
             MatrNr
          left outer join Vorlesungen v2 on v2.VorlNr = h2
             .VorlNr));
```

Durch die Verwendung von **with** und **case** wird die Anfrage übersichtlicher:

```
with GesamtSWS as (
  select sum(cast(SWS as decimal(5,2))) as AnzSWS
  from hoeren h2, Vorlesungen v2
  where v2.VorlNr = h2.VorlNr
),
GesamtStudenten as (
  select count(MatrNr) as AnzStudenten
  from Studenten
)
select s.*
from Studenten s
where s.MatrNr in (
  select h.MatrNr
  from hoeren h join Vorlesungen v on h.VorlNr = v.VorlNr
  group by h.MatrNr
  having sum(SWS) > (select AnzSWS / AnzStudenten
                     from GesamtSWS, GesamtStudenten));
```

Alternativ:

```
with SWSProStudent as (
select s.MatrNr,
  cast((case when sum(v.SWS) is null
             then 0 else sum(v.SWS)
        end) as real) as AnzSWS
 from Studenten s
   left outer join hoeren h on s.MatrNr = h.MatrNr
   left outer join Vorlesungen v on h.VorlNr = v.VorlNr
 group by s.MatrNr
)

select s.*
from Studenten s
where s.MatrNr in (select sws.MatrNr
                   from SWSProStudent sws
                   where sws.AnzSWS > (select avg(AnzSWS)
                                       from SWSProStudent)
                                          );
```

Hausaufgabe 2

Gegeben sei die folgende Relation `ZehnkampfD` mit Athletennamen und den von ihnen erreichten Punkten in den jeweiligen Zehnkampfdisziplinen:

ZehnkampfD : {Name, Disziplin, Punkte}

Name	Disziplin	Punkte
Eaton	100 m	450
Eaton	Speerwurf	420
...	...	...
Eaton	Weitsprung	420
Suarez	100 m	850
Suarez	Speerwurf	620
...	...	...

Finden Sie alle ZehnkämpferInnen, die in *allen* Disziplinen besser sind als der Athlet mit dem Namen *Bolt*. Formulieren Sie die Anfrage in SQL

a) mit korrelierter Unteranfrage

b) basierend auf Zählen

Sie dürfen davon ausgehen, dass jeder Sportler in jeder Disziplin angetreten ist.

Laden Sie zum Testen entweder die SQL-Datei von der Übungswebseite in ein lokal installiertes Datenbanksystem oder verwenden Sie die Webschnittstelle.

Lösung:

Mit korrelierter Unteranfrage:

```
select distinct a.Name
from ZehnkampfD as a
where not exists (
  select *
  from ZehnkampfD as a2
  where a2.Name = a.Name
    and exists (
      select *
      from ZehnkampfD as b
      where b.Disziplin = a2.Disziplin
        and b.Name = 'Bolt'
        and b.Punkte >= a2.Punkte
    )
)
```

Basierend auf Zählen

```
with besserAlsBolt(name, disziplin) as (
  select a.name, a.disziplin
  from zehnkampfd a, zehnkampfd b
  where b.name = 'Bolt'
    and a.disziplin = b.disziplin
    and a.punkte > b.punkte
),
disziplinen(anzahl) as (
  select count(distinct disziplin) as anzahl
  from zehnkampfd
)
select name
from besserAlsBolt
group by name
having count(*) = (select anzahl from disziplinen)
```

Hausaufgabe 3

Bringen Sie die folgende Relation verlustlos und abhängigkeitsbewahrend in die 3. NF.

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

FDs:

1. $AB \to CD$
2. $ABC \to D$
3. $E \to C$
4. $D \to C$
5. $CDE \to AB$

Beachten Sie, dass es für die Lösung notwendig ist, einen Kandidatenschlüssel zu ermitteln, jedoch nicht alle Kandidatenschlüssel. Beachten Sie außerdem, dass die Relation das Attribut F enthält, welches bei der Zerlegung nicht wegfallen darf.

Lösung:

Der Kandidatenschlüssel **muss** bestimmt werden, anders verliert man im dritten Schritt des Synthesealgorithmus potentiell Semantik und in diesem Fall insbesondere das Attribut F. Es müssen (da nicht explizit gefordert) jedoch nicht alle Kandidatenschlüssel sondern lediglich einer bestimmt werden. Ideales Verfahren hierzu ist, dass wir betrachten, welche Attribute nicht auf der rechten Seite von mindestens einer FD stehen. Diese **müssen** im Kandidatenschlüssel enthalten sein. Dies ist hier für E und F der Fall. Nun untersuchen wir, ob EF bereits ein Schlüssel für die Relation ist, womit wir dann bereits fertig wären. Wegen $AttrHuelle(FD, \{E, F\}) = \{E, F, C\} \neq R$ ist dies nicht der Fall. Wir versuchen nun möglichst intelligent ein Attribut hinzuzufügen, um Schlüsseleigenschaft zu erreichen. Da aus DE sofort AB folgt und E bereits definitiv im Schlüssel enthalten sein muss, fügen wir D hinzu. Damit gilt $AttrHuelle(FD, \{E, F, D\}) = \{E, F, D, C, A, B\} = R$, daher EDF ist Schlüssel. Insbesondere ist EFD auch Kandidatenschlüssel, da EF wie zuvor erklärt in jedem Schlüssel enthalten sein müssen, jedoch alleine kein Schlüssel sind, Reduktion nicht möglich, Schlüssel ist also Kandidatenschlüssel.

Ein weiterer Kandidatenschlüssel ist $ABEF$.

Jetzt muss der Synthesealgorithmus ausgeführt werden. Dazu wird zuerst die kanonische Überdeckung gebildet. Diese hat als ersten Schritt die Linksreduktion:

$$\begin{aligned} AB &\to CD \\ AB\not{C} &\to D \quad (AB \to C \text{ gilt}) \\ E &\to C \\ D &\to C \\ \not{C}DE &\to AB \quad (E \to C \text{ oder } D \to C \text{ gilt}) \end{aligned}$$

Danach wird die Rechtsreduktion durchgeführt:

$$\begin{aligned} AB &\rightarrow \not{C}\not{D} \\ AB\not{C} &\rightarrow D \\ E &\rightarrow C \\ D &\rightarrow C \\ \not{C}DE &\rightarrow AB \end{aligned}$$

C kann entfernt werden, da von D die FD $D \rightarrow C$ gilt. D kann entfernt werden, da $AB \rightarrow D$ durch die zweite FD gilt.

Nach Zusammenfassen erhält man für die kanonische Überdeckung also die folgenden FDs:

$$\begin{aligned} AB &\rightarrow D \\ E &\rightarrow C \\ D &\rightarrow C \\ DE &\rightarrow AB \end{aligned}$$

Daraus werden nun die folgenden Relationen erzeugt:

$$\begin{aligned} R_1 &: \{[A, B, D]\} \\ R_2 &: \{[C, E]\} \\ R_3 &: \{[C, D]\} \\ R_4 &: \{[A, B, D, E]\} \end{aligned}$$

Kein Kandidatenschlüssel ist in einer der Relationen enthalten. Darum erzeugen wir eine neue Relation für den Kandidatenschlüssel DEF. Da $\mathcal{R}_1 \subseteq \mathcal{R}_4$, kann R_1 entfernt werden. Das Ergebnis des Synthesealgorithmus ist dann:

$$\begin{aligned} R_2 &: \{[C, E]\} \\ R_3 &: \{[C, D]\} \\ R_4 &: \{[A, B, D, E]\} \\ R_\kappa &: \{[D, E, F]\} \end{aligned}$$

Hausaufgabe 4

Überführen Sie das folgende Schema verlustlos in die 4. NF:

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

$$\begin{aligned} AB &\rightarrow CDE \\ B &\rightarrow\!\!\!\!\rightarrow D \\ C &\rightarrow\!\!\!\!\rightarrow DE \end{aligned}$$

Beachten Sie, dass es zwei mögliche Lösungen gibt. Geben Sie beide an!

Lösung:

Dieses Schema halt AB als einzigen Kandidatenschlüssel. Dementsprechend verletzen beide MVDs die Bedingungen der 4. NF. Somit können beide zur Zerlegung verwendet werden. In diesem Schema führt das dann zu zwei verschiedenen Ergebnissen.

1. Lösung:

Für die erste Lösung verwenden wir die MVD $B \twoheadrightarrow D$ zur Dekomposition. Damit erhalten wir also:

$$\mathcal{R}_1 = \{\underline{B, D}\}$$
$$\mathcal{R}_2 = \{\underline{A, B}, C, E\}$$

Für $\mathcal{R}_1$ gelten jetzt keine (nicht-trivialen) FDs oder MVDs mehr, also ist es in 4. NF. Für $\mathcal{R}_2$ gelten $AB \to CE$ und $C \twoheadrightarrow AB$. Letztere ergibt sich aus der MVD $C \twoheadrightarrow DE$ mit Hilfe der Komplementregel. Also muss nun $\mathcal{R}_2$ Anhand von dieser MVD aufgeteilt werden. Daraus ergeben sich:

$$\mathcal{R}_{2.1} = \{\underline{A, B}, C\}$$
$$\mathcal{R}_{2.2} = \{\underline{C, E}\}$$

Für $\mathcal{R}_{2.1}$ gilt nur noch $AB \to C$. Für $\mathcal{R}_{2.2}$ gelten keine (nicht-trivialen) FDs oder MVDS. Damit sind beide in der 4. NF.

Das Ergebnis ist also: $\{\mathcal{R}_1, \mathcal{R}_{2.1}, \mathcal{R}_{2.2}\}$.

2. Lösung:

Für die zweite Lösung verwenden wir die MVD $C \twoheadrightarrow DE$ zur Dekomposition. Damit erhalten wir:

$$\mathcal{R}_1 = \{\underline{C, D, E}\}$$
$$\mathcal{R}_2 = \{\underline{A, B}, C\}$$

Für $\mathcal{R}_1$ gelten keine (nicht-trivialen) FDs oder MVDs mehr. Für $\mathcal{R}_2$ gilt $AB \to C$. Damit sind beide in der 4. NF und eine weitere Zerlegung ist nicht notwendig.

Das Ergebnis ist also: $\{\mathcal{R}_1, \mathcal{R}_2\}$.

Hausaufgabe 5

Fügen Sie die folgenden Tupel in eine anfangs leere erweiterbare Hashtabelle, welche 2 Einträge pro Bucket aufnehmen kann, ein. Dabei soll die Matrikelnummer als Suchschlüssel verwendet werden.

MatrNr	Name
2	Müller
8	Schmidt
19	Fischer
16	Huber
20	Bauer
34	Schneider
30	Wagner

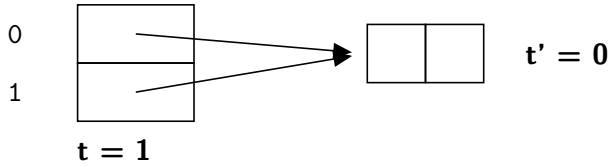
Verwenden Sie als Hashverfahren die inverse binäre Repräsentation der Matrikelnummer, wie in der Vorlesung beschrieben.

Lösung:

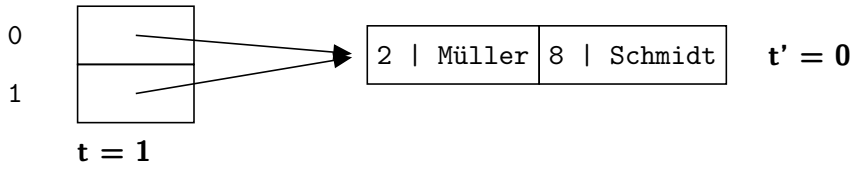
Zunächst errechnen wir den Hashwert für alle Studierende: Die inverse binäre Repräsentation der Matrikelnummer.

MatrNr	Name	Binär	Invers Binär
2	Müller	000010	010000
8	Schmidt	001000	000100
19	Fischer	010011	110010
16	Huber	010000	000010
20	Bauer	010100	001010
34	Schneider	100010	010001
30	Wagner	011110	011110

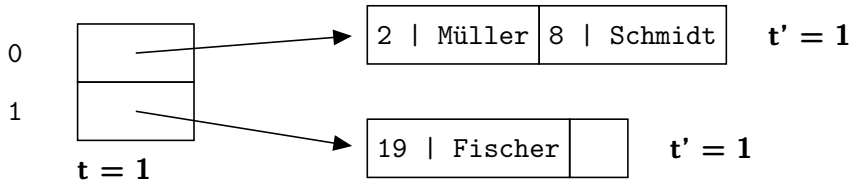
Zunächst eine leere erweiterbare Hashtabelle:



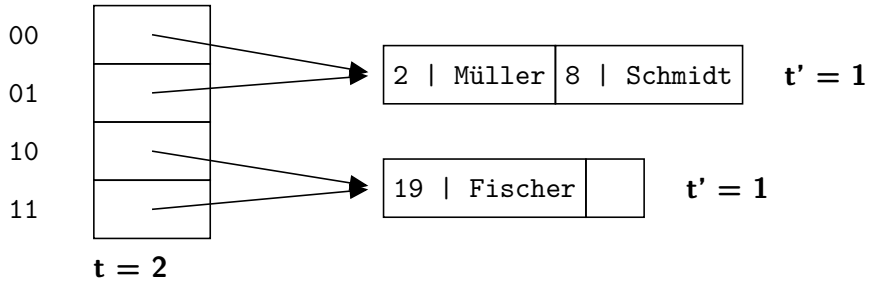
Wir fügen nun die ersten zwei Einträge ein, wonach die Hashtabelle wie folgt aussieht:



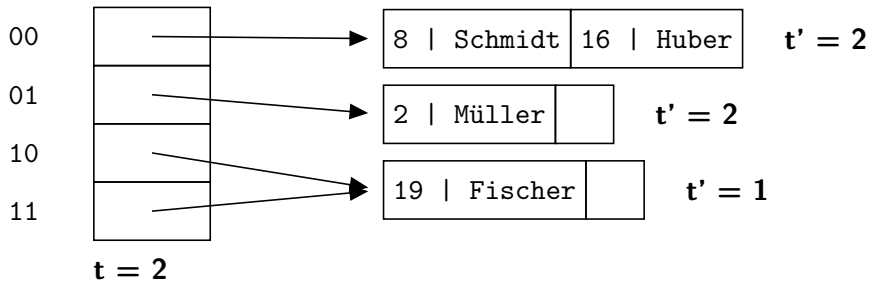
Der nächste Eintrag führt zu einem Überlauf. Da $t' < t$, können wir das Bucket teilen. Wir erhalten somit folgende Hashtabelle:



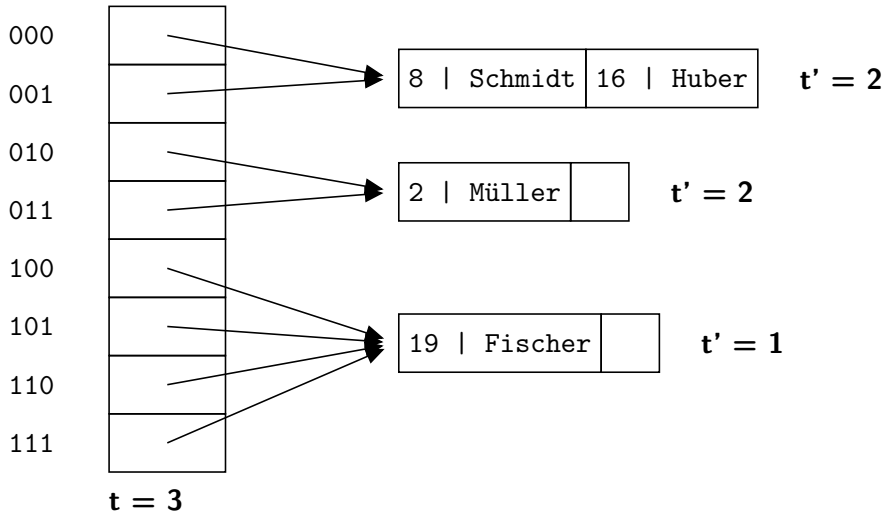
Der nächste Eintrag führt zu einem erneuten Überlauf! Leider in dem Bucket, für das $t = t'$ gilt. Wir müssen also die globale Tiefe erhöhen und erhalten somit folgende Hashtabelle:



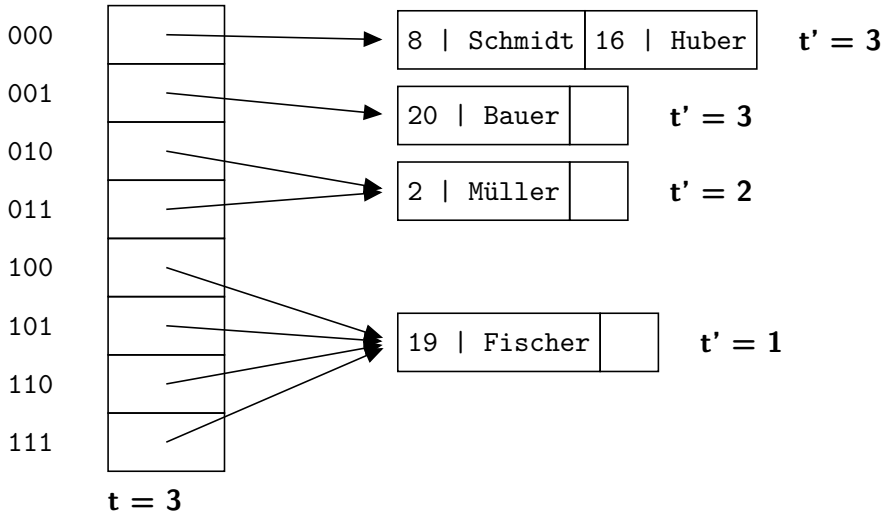
Nun können wir das erste Bucket splitten und unseren neuen Wert eintragen:



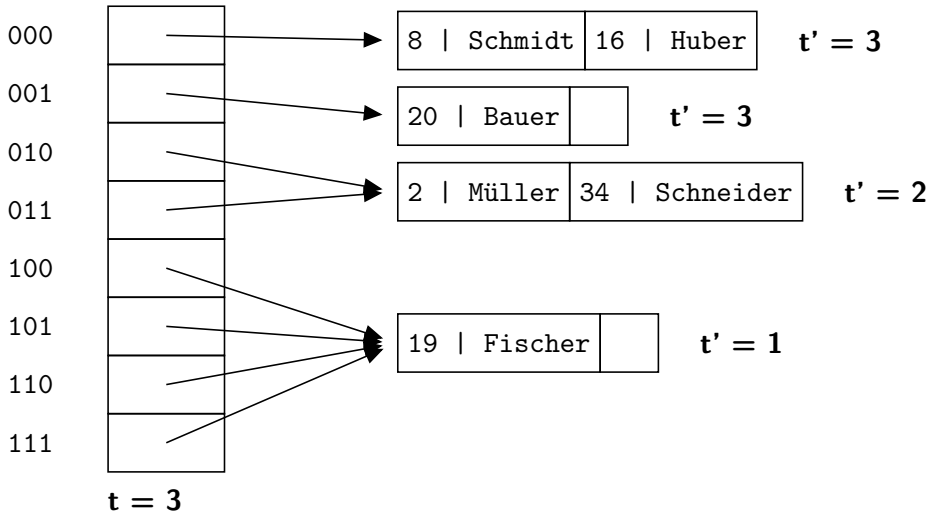
Bauer muss dem obersten Bucket hinzugefügt werden. Da dort kein Platz ist und $t = t'$ gilt, müssen wir erneut zuerst die globale Tiefe erhöhen.



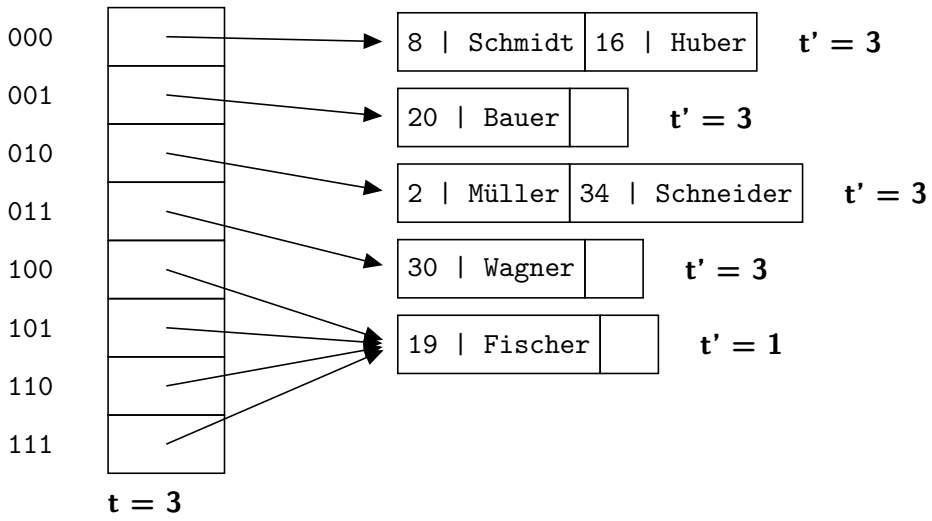
Wir können nun das oberste Bucket splitten, und Bauer hinzufügen:



Schneider passt in ein halbvolles Bucket:



Für Wagner müssen wir zuletzt noch einmal die lokale Tiefe erhöhen.


000
001
010
011
100
101
110
111
t = 3
8 | Schmidt
16 | Huber
t' = 3
20 | Bauer
t' = 3
2 | Müller
34 | Schneider
t' = 3
30 | Wagner
t' = 3
19 | Fischer
t' = 1