GDB Tutorium

Woche 06

Jigao Luo

TUM

28. November 2019

Wiederholung [1], [2]

- SQL: Structured Query Language.
  - Früherer Name: SEQUEL
  - SQL ist Multimenge (Bag) Semantik, aber relationale Algebra ist Menge Semantik
  - eine deklarative Anfragesprache (keine imperative Programmierung, auch keine funktionale)
  - deklarative Anfragesprache: ...*without describing its control flow*...
  - Einer der Gründe für deklarative Anfragesprache: Der Top User muss nur SQL lernen und eingeben. Das DBMS kümmert sich um Effizienz und Performanz.
  - DBMS: SQL compilation + SQL optimization + Indexing

[0] `https://en.wikipedia.org/wiki/Declarative_programming`

Wiederholung: case ... when ... end

```
select matrnr, (case when note <= 1.5 then 'sehr gut'
when note <= 2.5 then 'gut'
when note <= 3.5 then 'befriedigend'
when note <= 4.0 then 'ausreichend'
else 'nicht bestanden' end)
from pruefen;
```

Wiederholung: Modularisierung mit `with`

```
with table1(
select ... from ... where ... group by ... order by ...),
table2(
select ... from ... where ... group by ... order by ...)
select ... from ... where table1, table2;
```

Allquantor

1. Mit Existenzquantor umzuschreiben: Blatt 05
2. Mit `count(*)` zu vesuchen (für einfache Anfrage).

Die Studenten, die alle Vorlesungen hören.

Allquantor

1. Mit Existenzquantor umzuschreiben: Blatt 05
2. Mit `count(*)` zu vesuchen (für einfache Anfrage).

Die Studenten, die alle Vorlesungen hören.

```
select h.matrnr
from hoeren h
group by h.matrnr
having count(*) = (select count(*) from vorlesungen)
```

Die Studenten, die alle 4 SWS Vorlesungen hören.
Stimmt es?

```
select h.matrnr
from hoeren h
group by h.matrnr
having count(*) =
(select count(*) from vorlesungen where sws = 4)
```

Allquantor

Die Studenten, die alle 4 SWS Vorlesungen hören.

```
select h.matrnrwith hoerenStudentenWith4SWS(matrnr, vorlnr)
as( select h.matrnr, v.vorlnr
from hoeren h, vorlesungen v
where h.vorlnr = v.vorlnr and v.sws = 4)
select h.matrnr
from hoerenStudentenWith4SWS  h
group by h.matrnr
having count(*) =
(select count(*) from vorlesungen where sws = 4)
```

SQL Hausaufgabe

- push `https://github.com/cakebytheoceanLuo/IN0008_Tutorial`
- SQL per Email.
- Bevor push oder Email, gern test auf HyPer.

Hausaufgabe 2

“Bekanntheitsgrad”:
Formulieren Sie eine SQL-Anfrage, um den Bekanntheitsgrad von Studenten zu ermitteln. Gehen Sie dabei davon aus, dass Studenten sich aus gemeinsam besuchten Vorlesungen kennen. Sortieren Sie das Ergebnis absteigend nach Bekanntheitsgrad!

Mit `view` ⇔ Ohne `view`

Mit `view`

```
with bekannte as (select distinct h1.matrnr as student,
h2.matrnr as bekannter
from hoeren h1, hoeren h2
where h1.vorlnr = h2.vorlnr and h1.matrnr != h2.matrnr)
select s.name, s.matrnr, count(bekannter) as anzahl
from studenten s left outer join bekannte b
on s.matrnr = b.student group by s.name, s.matrnr order by anzahl desc;
```

Ohne `view`

```
select s.name, s.matrnr, count(bekannter) as anzahl
from studenten s left outer join (
select distinct h1.matrnr as student, h2.matrnr
as bekannter
from hoeren h1, hoeren h2
where h1.vorlnr = h2.vorlnr and h1.matrnr != h2.matrnr)
as b
on s.matrnr = b.student group by s.name, s.matrnr order by anzahl desc;
```

Hausaufgabe 3

a) Geben Sie alle Einträge des Fahrplans aus, deren Abfahrtshaltestelle das Wort “Garching” **enthält**.

Hausaufgabe 3

a) Geben Sie alle Einträge des Fahrplans aus, deren Abfahrtshaltestelle das Wort “Garching” **enthält**.

String **like wildcards**: Übung Slide 05

Hausaufgabe 3

b) Geben Sie alle Einträge des Fahrplans mit dem zusätzlichen Attribut “Verkehrsmittel” aus. Alle Linien, die mit “U” anfangen, haben das Verkehrsmittel “U-Bahn”, alle die mit “S” anfangen “S-Bahn”, und alle restlichen “Bus/Tram”.

Hausaufgabe 3

c) Finden Sie alle Abfahrten ab Garching, “Forschungszentrum”, die Sie heute noch erreichen können. Sortieren Sie das Ergebnis aufsteigend nach Abfahrtszeit.

Hausaufgabe 3

d) Finden Sie alle Fahrten zwischen zwei Haltestellen (nicht-transitiv), die mindestens drei und höchstens fünf Minuten dauern.

Bonus: Berücksichtigen Sie, dass Fahrten über Mitternacht möglich sind, z.B. Abfahrt um 23:58 Uhr und Ankunft um 00:02 Uhr.

Hausaufgabe 4

Formulieren Sie die folgende Anfrage auf dem bekannten Unischema in SQL: Ermitteln Sie für jede Vorlesung, wie viele Studenten diese vorgezogen haben. Ein Student hat eine Vor- lesung vorgezogen, wenn er in einem früheren Semester ist als der “Modus” der Semester der Hörer dieser Vorlesung. Der Modus ist definiert als der Wert, der am häufigsten vor- kommt – für diese Anfrage also das Semester, in dem die meisten Hörer dieser Vorlesung sind. Falls es mehrere Semester dieser Art gibt, soll nur das niedrigste zählen.

Beachten Sie, dass auch Vorlesungen ohne Hörer, sowie Vorlesungen deren Hörer alle im gleichen Semester sind, ausgegeben werden sollen.

Geben Sie für jede Vorlesung die Vorlesungsnummer, den Titel und die Anzahl der “Vorzieher” aus.

Hausaufgabe 4: Schritten

1. Für jede Vorlesung bestimmen, von wie vielen Stundenten sie pro Semester gehört wird.
2. Den Modus der Vorlesung bestimmen. (mit `max`)
3. Joins mit Bedienung: wenn er in einem früheren Semester ist als der “Modus” der Semester der Hörer dieser Vorlesung.

vorlnr	semester	anzahl
4052	6	1
5001	10	1
5001	2	2
5001	6	1
5022	2	1
5022	12	1
5041	3	1
5041	2	1
5049	2	1
5052	3	1

1

vorlnr	modus
5216	3
5001	2
5041	2
4052	6
5049	2
5259	3
5022	2
5052	3

2

vorlnr	titel	anzahl
5216	Bioethik	0
4630	Die 3 Kritiken	0
5001	Grundzüge	0
5043	Erkenntnistheorie	0
5041	Ethik	0
4052	Logik	0
5049	Mäeutik	0
5259	Der Wiener Kreis	0
5022	Glaube und Wissen	0
5052	Wissenschaftstheorie	0

3

vorl_modus Mit Subquery ⇔ Ohne Subquery

Mit Subquery

```
with vorl_semester_anz as ( select h.vorlnr, s.semester, count(*) as anzahl
from hoeren h, Studenten s where h.matrnr = s.matrnr group by
h.vorlnr, s.semester
), vorl_modus as (select v1.vorlnr, min(v1.semester) as
modus from vorl_semester_anz v1
where v1.anzahl = (select max(v2.anzahl) from
vorl_semester_anz v2 where v1.vorlnr = v2.vorlnr) group by
v1.vorlnr)
```

Ohne Subquery

```
with vorl_semester_anz as ( /*gleich wie oben*/
) max_anzahl as (
select vorlnr, max(v.anzahl) as max from vorl_semeste_anz v
group by vorlnr
) vorl_modus as (select v.vorlnr, min(v.semester) as modus
from vorl_semester_anz v, max_anzahl m where
v.anzahl = m.max and v.vorlnr = m.vorlnr group by v.vorlnr)
```

Mit Subquery $\Leftrightarrow$ Ohne Subquery

Mit Subquery

```
for vorl_semester_anz v1:
        for vorl_semester_anz v2:
                if (v1.vorlnr = v2.vorlnr)
                        return max(v2.anzahl);
```

Kann noch schlimmer: wenn die gleiche Subquery-Tabelle von v2 für jede v1 generiert werden.
$\Rightarrow$ Wenn Query Optimizer “Unnesting [3]” nicht kann, dann $O(n^2)$

Ohne Subquery

```
        max_anzahl m;
        vorl_semeste_anz v;
        m ⋈ v;
```

Join ist effizienter. z.B: Sort-Merge-Join:
$O(nlogn) \Rightarrow O(|m|log|m| + |v|log|v| + |JoinResult|)$


T. Neumann, “Grundlagen: Datenbanken, Kapitel 04”,,


A. Kemper, “Grundlagen: Datenbanken, Kapitel 04”,,


T. Neumann und A. Kemper, “Unnesting Arbitrary Queries”,,