GDB Tutorium

Woche 05

Jigao Luo

TUM

21. November 2019

Wiederholung [1]

- 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

SQL üben:

- HyPer Web-interface: a single thread on a low-end server
- postgresql (local laufen)

Wiederholung:
Select Klausel = Projektion in Relationen Algebra

```
select distinct s.Name
from Studenten s;
```

Wiederholung:
Select Klausel = Projektion in Relationen Algebra

```
select distinct s.Name
from Studenten s;
```

distinct: Duplikat eliminiert

```
select distinct s.Name
from Studenten s;
```

Wiederholung:
Select Klausel = Projektion in Relationen Algebra

```
select distinct s.Name
from Studenten s;
```

distinct: Duplikat eliminiert

```
select distinct s.Name
from Studenten s;
```

*: alle Attribute

```
select *
from Studenten s;
```

Wiederholung: Where Klausel mit Prädikaten = Selection in Relationen Algebra

```
select distinct s.Name
from Studenten s
where semester >= 2 and semester <= 4;
```

Wiederholung: Where Klausel mit Prädikaten = Selection in Relationen Algebra

```
select distinct s.Name
from Studenten s
where semester >= 2 and semester <= 4;
```

between:

```
select distinct s.Name
from Studenten s
where semester between 2 and 4;
```

Wiederholung: Where Klausel mit Prädikaten = Selection in Relationen Algebra

String

```
select *
from Studenten s
where name = 'Mustermann';
```

Wiederholung: Where Klausel mit Prädikaten = Selection in Relationen Algebra

String

```
select *
from Studenten s
where name = 'Mustermann';
```

like wildcards: _ für EIN biliebiges Zeichen, % für eine beliebige Zeichenkette (Länger ≥ 0)

```
select *
from Studenten s
where name like 'M%';
```

```
select *
from Studenten s
where name like 'Musterman_';
```

Join R1 on R2

```
select *
from R1 [cross|inner|natural|left outer|
right outer|full outer] join R2 [on R1.A = R2.B];
```

Join R1 on R2

```
select *
from R1 [cross|inner|natural|left outer|
right outer|full outer] join R2 [on R1.A = R2.B];
```

Das Keyword “Join” ist nicht immer nötig.

```
select *
from Student S, hoeren B, Vorlesung V
where S.MatrNr = B.MatrNr and B.Nr = V.Nr;
```

Join R1 on R2

```
select *
from R1 [cross|inner|natural|left outer|
right outer|full outer] join R2 [on R1.A = R2.B];
```

Das Keyword “Join” ist nicht immer nötig.

```
select *
from Student S, hoeren B, Vorlesung V
where S.MatrNr = B.MatrNr and B.Nr = V.Nr;
```

Bei “Join” muss nicht immer ein Key als Prädikaten eingeben werden. Eine SINNLOSE Anfrage als ein Beispiel:

```
select *
from vorlesungen v, studenten s
where v.sws = s.semester;
```

Order by: Sortierung

Tupel in einer Relation sind nicht (automatisch) sortiert, wenn nichts explizit anzufordern.
Wenn Anfrage parallel ausgefürt werden, kann sein, die Reihenfolge ist nicht jedes mal identisch.

aufsteigend:

```
select *
from Student S
order by semester;
```

absteigend:

```
select *
from Student S
order by semester desc;
```

Group by: Gruppieren

```
select vorlnr, count(*) as anzahl
from hoeren
order by vorlnr;
```

Alle Attribute die nicht in der group by-Klausel auftauchen dürfen nur aggregiert in der select-Klausel stehen

falsch, da nach der Gruppierung existiert keine Column: matrnr.

```
select vorlnr, matrnr
from hoeren
order by vorlnr;
```

SQL Hausaufgabe

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

Hausaufgabe 2

Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL. Geben Sie alle Ergebnisse **duplikatfrei** aus.

(a) Finden Sie die Studenten, die Sokrates aus Vorlesung(en) kennen.

Hausaufgabe 2

Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL. Geben Sie alle Ergebnisse **duplikatfrei** aus.

(b) Finden Sie die Studenten, die Vorlesungen hören, die auch Fichte hört.

Hausaufgabe 2

Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL. Geben Sie alle Ergebnisse **duplikatfrei** aus.

(c) Finden Sie die Assistenten von Professoren, die den Studenten Carnap unterrichtet haben – z.B. als potentielle Betreuer seiner Bachelorarbeit.

Hausaufgabe 2

Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL. Geben Sie alle Ergebnisse **duplikatfrei** aus.

(d) Geben Sie die Namen der Professoren an, die Theophrastos aus Vorlesungen kennt.

Hausaufgabe 2

Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL. Geben Sie alle Ergebnisse **duplikatfrei** aus.

(e) Welche Vorlesungen werden von Studenten im Bachelorstudium (1. – 6. Semester) gehört? Geben Sie die Titel dieser Vorlesungen an.

Hausaufgabe 2

Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL. Geben Sie alle Ergebnisse **duplikatfrei** aus.

(f) Bestimmen Sie für jede Vorlesung wie viele Studenten diese hören. Geben Sie auch Vorlesungen ohne Hörer aus. Sortieren Sie das Ergebnis absteigend nach Anzahl der Hörer.

Hausaufgabe 3

Bestimmen Sie für alle Studenten eine gewichtete Durchschnittsnote ihrer Prüfungen.
Die Gewichtung der einzelnen Prüfungen erfolgt gemäß dem Vorlesungsumfang (SWS).
Dies entspricht dem Verfahren der Durchschnittsnotenberechnung für Ihr Bachelor-Zeugnis.

Hausaufgabe 4

Folgender Ausdruck im Tupelkalkül gibt alle Studenten aus, die alle von ihnen gehörten Vorlesungen bestanden haben.

$\{s \mid s \in \text{Studenten} \wedge \forall\, h \in \text{hoeren}(h.\text{MatrNr} = s.\text{MatrNr} \Rightarrow \exists\, p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$

Übersetzen Sie diese Anfrage nun in SQL. Da SQL keine Allquantoren und Implikationen unterstützt, müssen Sie sie dazu zunächst umformen.

a) Formen Sie den Ausdruck in einen Äquivalenten um, der keine Implikationen oder Allquantoren verwendet.

$A \Rightarrow B \Leftrightarrow \neg A \bigvee B$
$\forall x(P(x)) \Leftrightarrow \neg \exists x(\neg P(x))$

Hausaufgabe 4

Folgender Ausdruck im Tupelkalkül gibt alle Studenten aus, die **ALLE von ihnen gehörten** Vorlesungen bestanden haben.

$$\{s \mid s \in \text{Studenten} \wedge \forall\, h \in \text{hoeren}(h.\text{MatrNr} = s.\text{MatrNr} \Rightarrow \exists\, p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$$

Übersetzen Sie diese Anfrage nun in SQL. Da SQL keine Allquantoren und Implikationen unterstützt, müssen Sie sie dazu zunächst umformen.

b) Übersetzen Sie den so erlangten Ausdruck in SQL. Testen Sie ihn in der Webschnitt- stelle.

T. Neumann, *Grundlagen: Datenbanken, Kapitel 04*.