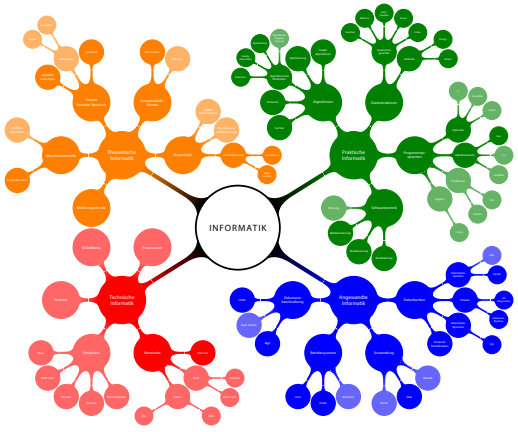
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

Kapitel 38

SQL – Einfache Anfragen

Die strukturierte Anfragesprache

Vorlesung Einführung in die Informatik 2 vom 3. Juni 2014 von Till Tantau



Lernziele von Kapitel 38

1. SQL-Anfragen an eine Datenbank formulieren können
2. Einfache SQL-Joins benutzen können
3. SQL-Funktionen benutzen können

Gliederung von Kapitel 38

38.1 Einfache Anfragen

38.1.1 Grundaufbau

38.1.2 Verbesserte Ausgaben

38.1.3 Joins

38.2 Anfragen für Fortgeschrittene

38.2.1 Prädikate

38.2.2 Funktionen

Formulierung einer Anfrage in SQL.

38.1 Einfache Anfragen
▶ Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

- Anfragen haben generell folgende Form:

```
select A_1,...,A_n
from R_1,...,R_m
where P
```

  Hierbei sind
  - A_1 bis A_n Attribute,
  - R_1 bis R_m Relationen und
  - P ein Prädikat.
- Der Effekt dieser Anfrage ist es, die folgende Relation auszugeben:

$$\pi_{A_1,\ldots,A_n}(\sigma_P(R_1 \times \cdots \times R_m))$$

Beispiele einfacher Anfragen.

38.1 Einfache Anfragen
▶ Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

- Welche schwarzen Schafe gibt es?

```
mysql> select name
         from schaf
         where farbe = "schwarz";
+----------+
| name     |
+----------+
| Flauschi |
| Peter    |
+----------+
```

- Welche schwarzen Schafe gibt es und welches Fell haben sie?

```
mysql> select name, fell
         from schaf
         where farbe = "schwarz";
+----------+-----------+
| name     | fell      |
+----------+-----------+
| Flauschi | wuschelig |
| Peter    | wuschelig |
+----------+-----------+
```

Beispiele einfacher Anfragen.

38.1 Einfache Anfragen
▶ Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

- Welche Ställe haben eine Größe von mindestens 50?

```
mysql> select nummer
         from stall
         where groesse >= 50;
+--------+
| nummer |
+--------+
|      1 |
+--------+
```

- Welche Stall-Entitäten haben eine Größe von mindestens 50?

```
mysql> select *
         from stall
         where groesse >= 50;
+--------+----------+
| nummer | groesse |
+--------+----------+
|      1 |       50 |
+--------+----------+
```

Beschränkung der Zeilenanzahl.

38.1 Einfache Anfragen
Grundaufbau
▶ Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

- Möchte man wissen, was beispielsweise alles in der Tabelle `stall` steht, so geht dies sehr leicht mit folgender Anfrage:

```
mysql> select * from stall;
+--------+---------+
| nummer | groesse |
+--------+---------+
|      1 |      50 |
|      5 |      25 |
+--------+---------+
```

- Bei großen Tabellen (wie denen von Ensembl mit vielen Megabyte Inhalt), dauert dies aber viel zu lange.
- Der Trick ist nun, hinter einen **`select`**-Befehl eine *Limitierung* zu schreiben:

```
mysql> select * from stall limit 5;
```

Dies sorgt dafür, dass nur die ersten 5 Zeilen ausgegeben werden.

Beschränkung der Strings.

- Man kann statt einem gesamten String wie einem Namen oder einer DNA-Sequenz auch nur einen Teil ausgeben lassen.
- Dazu schreibt man statt dem Attribut einfach `substring(n,s,l)`, wobei *n* der Name des Attributs ist, *s* das Anfangszeichen im String (mit 1 beginnend) und *l* die Länge des gewünschten Teilstrings ist.


```
mysql> select substring(name,1,3), farbe, fell from stall;
+--------------------+---------+-----------+
| substring(name,1,3) | farbe   | fell      |
+--------------------+---------+-----------+
| Dol                | weiss   | lockig    |
| Max                | schwarz | wuschelig |
| Pet                | schwarz | wuschelig |
+--------------------+---------+-----------+
mysql> select substring(name,2,3), farbe, fell from stall;
+--------------------+---------+-----------+
| substring(name,2,3) | farbe   | fell      |
+--------------------+---------+-----------+
| oll                | weiss   | lockig    |
| ax                 | schwarz | wuschelig |
| ete                | schwarz | wuschelig |
+--------------------+---------+-----------+
```

Sortierung der Ausgabe.

- Oft möchte man die Ausgabe *sortieren*. (Normalerweise werden die Daten in einer nicht vorhersagbaren Reihenfolge ausgegeben.)
- Dazu kann man nach einem `select`-Befehl ein *Sortierungsattribut* angeben:

```
mysql> select * from stall order by groesse;
+--------+---------+
| nummer | groesse |
+--------+---------+
|      5 |      25 |
|      1 |      50 |
+--------+---------+
```

- *Nach* dem Attribut kann man noch `desc` angeben, um die Reihenfolge umzudrehen.

```
mysql> select * from stall order by nummer desc;
+--------+---------+
| nummer | groesse |
+--------+---------+
|      5 |      25 |
|      1 |      50 |
+--------+---------+
```


38.1 Einfache Anfragen
Grundaufbau
▶ Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

5-Minuten-Aufgabe

38.1 Einfache Anfragen
Grundaufbau
▶ Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

Zur Übung

Geben Sie den SQL-Befehl an, mit dessen Hilfe Sie die fünf größten Ställe angezeigt bekommen.

Zur Erinnerung

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
▶ Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

- Will man Informationen aus zwei Relationen *zusammenfügen*, so benötigt man ein *Kreuzprodukt*, gefolgt von einer *Filterung*.
- Bei der Filterung streicht man alle Zeilen, in der zwei bestimmte gleichbenannte Attribute nicht gleich sind.
- Eine solche Verbindung nennt man auch einen *Join* anhand dieses Attributs.

Beispiele einer Kreuzproduktes.

Das Kreuzprodukt von Schafen und der Liegt-In-Relation.

```
mysql> select * from schaf, liegt_in;
+----------+---------+-----------+----------+--------+
| name     | farbe   | fell      | name     | nummer |
+----------+---------+-----------+----------+--------+
| Dolly    | weiss   | lockig    | Dolly    |      1 |
| Dolly    | weiss   | lockig    | Flauschi |      5 |
| Flauschi | schwarz | wuschelig | Dolly    |      1 |
| Flauschi | schwarz | wuschelig | Flauschi |      5 |
| Peter    | schwarz | wuschelig | Dolly    |      1 |
| Peter    | schwarz | wuschelig | Flauschi |      5 |
+----------+---------+-----------+----------+--------+
```

Die Ställe der schwarzen Schafe.

Die Anfrage $\pi_{\text{Stall-Nummer}}\left(\sigma_{\text{Name}=\text{Schaf-Name}}\left(\sigma_{\text{Farbe}=\text{schwarz}}(\text{Schaf}) \times \text{Schläft-in}\right)\right)$ in SQL:

```
mysql> select nummer
         from schaf, liegt_in
         where
           farbe       = "schwarz" and
           schaf.name  = liegt_in.name;
+--------+
| nummer |
+--------+
|      5 |
+--------+
```

Eine komplexe Anfrage.

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
▶ Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

Welche Größe haben die Ställe der Schafe?

```
mysql> select schaf.name, groesse
         from schaf, liegt_in, stall
         where
           schaf.name   = liegt_in.name and
           stall.nummer = liegt_in.nummer;
+----------+---------+
| name     | groesse |
+----------+---------+
| Dolly    |      50 |
| Flauschi |      25 |
+----------+---------+
```

5-Minuten-Aufgabe

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
▸ Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

Zur Übung

Nehmen wir an, die folgenden Befehle wurden ausgeführt und die Tabellen gefüllt.

```
create table schaf     (name char(20),
                        farbe char(20),
                        fell char(20));
create table liegt_in (name char(20), nummer integer);
create table stall     (nummer integer, groesse float);
create table mama      (name char(20), mutter char(20));

insert into ...;
```

1. Geben Sie einen SQL-Befehl an, um die Größe der Ställe zu bestimmen, in denen braune Schafe liegen.
2. Geben Sie einen SQL-Befehl an, um den Stall zu bestimmen, in dem die Mutter von Dolly schläft.

Erste Vereinfachung der Syntax bei Joins.

Da Joins so oft vorkommen, gibt es eine spezielle Syntax hierfür: Der Befehl

```
select ... from tabelle1 join tabelle2
                         using (A1, ..., An)
```

hat denselben Effekt wie

```
select ... from tabelle1, tabelle2
           where tabelle1.A1 = tabelle2.A1 and
                 ...
                 tabelle1.An = tabelle2.An and
```

Das Beispiel von oben schreibt sich dann so:

```
mysql> select schaf.name, groesse
         from schaf join liegt_in using (name)
                    join stall    using (nummer);
+----------+---------+
| name     | groesse |
+----------+---------+
| Dolly    |      50 |
| Flauschi |      25 |
+----------+---------+
```

Zweite Vereinfachung der Syntax bei Joins.

Beim Join von zwei Tabellen ist es »natürlich«, die Tabellen gerade anhand derjenigen Spalten zu »joinen«, die in beiden Tabellen vorkommen. Diesen `natural join` kann man direkt hinschreiben:

```
select ... from tabelle1 natural join tabelle2
```

hat denselben Effekt wie

```
select ... from tabelle1, tabelle2
           where tabelle1.A1 = tabelle2.A1 and
                 ...
                 tabelle1.An = tabelle2.An and
```

wobei `A1` bis `An` die Spalten sind, die in beiden Tabellen vorkommen.
Das Beispiel von oben schreibt sich dann so:

```
mysql> select name, groesse
         from schaf natural join liegt_in
                    natural join stall;
+----------+---------+
| name     | groesse |
+----------+---------+
| Dolly    |      50 |
| Flauschi |      25 |
+----------+---------+
```


38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
▶ Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
Funktionen

Wie lautet die Prädikatsyntax?

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
▶ Prädikate
Funktionen

- Prädikate folgen in SQL dem Schlüsselwort **`where`**.
- Sie sind (ähnlich wie bei Java) boolesche Ausdrücke.
- Die Variablen in einem Prädikat sind gerade die Attribute einer Zeile. (Haben zwei Relationen `R_1` und `R_2` das gleiche Attribut `A`, so muss man `R_1.A` oder `R_2.A` schreiben.)
- Die Hauptunterschiede zu Java-Ausdrücken:
  1. Gleichheit wird mit `=` statt mit `==` geprüft.
  2. Strings lassen sich mittels `<`, `<=` usw. lexikographisch vergleichen.
  3. Statt `!`, `||` und `&&` kann man (und tut man) auch **`not`**, **`or`** und **`and`** schreiben.
  4. Mit **`like`** und **`regexp`** kann man komplexe Textvergleiche anstellen (dazu gleich mehr).

Beispiele für Prädikate.

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
▶ Prädikate
Funktionen


```
mysql> select ... where
    farbe = "schwarz" and
    not(nummer < 5);
...
mysql> select ... where
  liegt_in.nummer = stall.nummer and
  not(farbe like "wei%");
...
```

Die drei Arten von Stringvergleichen in SQL.

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
▶ Prädikate
Funktionen

In SQL gibt es drei Methoden, Strings zu vergleichen:

1. Ein *direkter lexikographischer Vergleich* mittels = oder >=.
2. Eine Art *Ähnlichkeitsvergleich* mittels `a like b`. Dieser Vergleich ist wahr, falls:
   - 2.1 Die Strings `a` und `b` gleich sind, wobei aber
   - 2.2 für jedes Vorkommen des Unterstrichs in `b` an der entsprechenden Stelle in `a` ein beliebiges Zeichen stehen darf und
   - 2.3 für jedes Vorkommen eines Prozentzeichens in `b` an der entsprechenden Stelle in `a` eine beliebige (auch leere) Zeichenfolge stehen darf.

   Ist beispielsweise `b` der String `H_llo`, so könnte `a` sowohl `Hallo` als auch `Hello` sein, nicht aber `Haallo`. Ist `b` der String `H%llo`, so könnte `a` nun `Haallo` sein, nicht aber `Halo`.
3. Ein Vergleich mit einem *regulären Ausdruck.*

Anwendung von regulären Ausdrücke in SQL.

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
▶ Prädikate
Funktionen

Beispiel

Wir wollen in einer DNA-Sequenz nach einem Startkodon, gefolgt von beliebigen Basentriplets, gefolgt von einem Stoppkodon suchen. Zwei mögliche Schreibweisen für den Ausdruck:

1. `atg((a|t|g|c)(a|t|g|c)(a|t|g|c))*(tag|taa|tga)`
2. `atg([atgc]{3})*(tag|taa|tga)`

Anwendung in SQL

```
mysql> select * from dna
                where sequence regexp
                    "atg([atgc]{3})*(tag|taa|tga)"
```

5-Minuten-Aufgabe

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
▶ Prädikate
Funktionen

Zur Übung

Geben Sie einen SQL-Befehl an, der aus der Tabelle `dna` alle Zeilen auswählt, bei denen das Attribut `sequence` alle der folgenden Eigenschaften hat:

1. Es kommt irgendwo `agu` vor.
2. Es kommt irgendwo `ccc` vor und später `aaa`.
3. Der String endet nicht mit `aaa`.

Neu: Zeilenweise Anwendung statt Projektionen.

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
▶ Funktionen

Definition

Seien $A \subseteq A_1 \times \cdots \times A_n$ eine Relation und seien $f_i : A_1 \times \cdots \times A_n \to B_i$ Funktionen. Dann ist die *zeilenweise Anwendung* dieser Funktionen wie folgt definiert:

$$\begin{aligned}\zeta_{f_1,\ldots,f_m}(A) &= \{(f_1(a),\ldots,f_m(a)) \mid a \in A\} \\ &\subseteq B_1 \times \cdots \times B_m.\end{aligned}$$

Beispiel

$\zeta_{\text{erstes Zeichen von(Name)},\text{Länge(Farbe)+Länge(Fell)}}(\text{Schaf}) =$

B_1	B_2
D	10
F	16
P	16

Die zeilenweise Anwendung in SQL.

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
▶ Funktionen

- Bei einem Select-Befehl kann man statt Attributen auch beliebige Funktionen angeben.
- Tatsächlich sind die normalen Projektionen auch nur Spezialfälle der zeilenweise Anwendung einer (sehr einfachen) Funktion.
- Welche Funktionen möglich sind, muss man in der Dokumentation nachschlagen. Einige Beispiele: `abs` für den Absolutbetrag ein Zahl, `sin` für den Sinus einer Zahl, `length` für die Länge eines Strings, **`substring`** für einen Teilstring, `concat` für die Verkettung mehrerer Strings.

```
mysql> select
  substring(name,1,1), length(farbe)+length(fell)
  from schaf;
```

Zusammenfassung

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
▶ Funktionen

Grundsyntax von Anfragen

Anfragen haben generell folgende Form:

```
select A_1,...,A_n
from R_1,...,R_m
where P
```

Hierbei sind

- A_1 bis A_n Attribute,
- R_1 bis R_m Relationen und
- P ein Prädikat.

Zusammenfassung

38.1 Einfache Anfragen
Grundaufbau
Verbesserte Ausgaben
Joins
38.2 Anfragen für Fortgeschrittene
Prädikate
▶ Funktionen

Join-Syntax

Joins kann man besonders einfach schreiben:

```
select A_1,...,A_n
from R_1 natural join R2 natural join R3 ...
where P
```

Dies liefert den Join der Relationen anhand der Spalten, die jeweils die Relationen mit den vorherigen gemeinsam haben.

Funktionen statt Projektionen

Man kann statt Projektionen auch beliebige Funktionen zeilenweise anwenden.