

# DATENBANKEN & SQL

– Einfache Abfragen –

Kurs am RRZK der Universität zu Köln

Rüdiger Voigt, M.A.

22.07. – 26.07.2019

# Übersicht

- 1 Daten auswählen
  - SELECT
  - WHERE
  - Sonderfall NULL
  - ORDER BY
  - LIMIT
  - CONCAT
- 2 AS
- 3 Berechnungen
  - Aggregatfunktionen
- 4 Views

# Konventionen in SQL

- Sie können eine Query durch Zeilenumbrüche übersichtlicher gestalten. Sie muss aber immer mit einem Semikolon (;) abgeschlossen werden.
- Kommandos werden in GROßBUCHSTABEN geschrieben.

# Daten auswählen

# SELECT Statements

Ein SELECT-Statement ist der grundlegende Befehl um Daten aus einer Datenbank abzufragen. In seiner Grundform besteht es aus:

- 1 Dem Befehl SELECT,
- 2 einer durch Kommata getrennten Liste von Columns,
- 3 dem Keyword FROM gefolgt vom Namen der abgefragten Tabelle.

Beispiel:

```
SELECT a, b, c FROM x;
```

gibt die Spalten a, b und c der Tabelle x zurück. Die Reihenfolge der Spalten können Sie nach Belieben variieren indem Sie diese innerhalb der Auflistung umsortieren.

# SELECT Statements II

## Warnung

**SELECT \* FROM table;**

ist ein beliebter Shortcut um ALLE Spalten einer Tabelle abzufragen.

Damit verbunden sind zwei Probleme:

- 1 Code, der auf die Reihenfolge oder eine feste Anzahl der Spalten angewiesen ist, funktioniert nach Änderungen der Datenbankstruktur nicht mehr.
- 2 Je nach Datenbank ziehen Sie unnötig einige hundert Spalten, das heißt Sie mindern die Performance des Servers, des Netzwerks und Ihrer Anwendung!

# SELECT Statements III

Das Ergebnis Ihrer Abfrage mit SELECT (das Resultset) ist flüchtig. Wenn Sie es mit dem Aufruf nicht in irgendeiner Form speichern, ist das Ergebnis nach der Anzeige wieder weg.

Wenn Sie das Ergebnis wieder brauchen, geben Sie die Abfrage einfach erneut ein. Sollten sich die Daten im System geändert haben, erhalten Sie (je nachdem) ein anderes Ergebnis.

Die Ausgabe des SELECT Statement können Sie nicht benutzen um Daten zu editieren.

# WHERE I

## **WHERE ist die wichtigste Ergänzung eines SELECT-, UPDATE oder DELETE-Statement!**

- Mittels WHERE formulieren Sie eine oder mehrere Bedingungen, welche Zeilen erfüllen müssen um Teil des Resultset zu werden.
- WHERE ist extrem vielseitig. MariaDB Docs: “In the WHERE clause, you can use any of the functions and operators that MariaDB supports, except for aggregate (summary) functions.”
- Nur mit WHERE und einem Unique Identifier können Sie einen ganz bestimmten Eintrag auswählen.



# WHERE II

Auswahl der Operatoren, die Ihnen in der WHERE Clause zur Verfügung stehen:

- = gleich (*nicht == wie in vielen Programmiersprachen*)
- != ungleich
- **AND** Verknüpfung
- **OR** inklusives Oder (A oder B oder beide)
- < <= > >= kleiner, kleiner gleich, größer und größer gleich

## WHERE III – Kombination von Operatoren

Sie können in einer WHERE Clause mehrere Operatoren verwenden. Zum Beispiel:

```
SELECT firstName, lastName  
FROM customers  
WHERE creditRating = "A"AND revenue2015 > 1000000  
AND revenue2016 > 500000;
```

Eine Liste aller Kunden, deren Kreditwürdigkeit mit A bewertet wird und die darüber hinaus im Jahr 2015 mehr als eine Million sowie im Jahr 2016 bereits mehr als eine halbe Million Umsatz generiert haben.

## WHERE IV – Verwendung von Klammern

Mit Klammern können Sie Bedingungen zusammenfassen. Zum Beispiel:

```
SELECT firstName, lastName  
FROM customers  
WHERE creditRating = "A"  
OR (revenue2015 > 1000000 AND revenue2016 > 500000)  
OR vip = 1;
```

Eine Liste aller Kunden, welche entweder über eine Kreditwürdigkeit der Stufe A verfügen, oder beide Umsatzziele erfüllen, oder als VIPs markiert sind, oder aber alle drei Kriterien erfüllen.

# Sonderfall NULL

# Sonderfall NULL-Werte I

## Warnung

- 1 NULL signalisiert eine leere Zelle oder ein leeres Ergebnis. NULL hat nichts mit dem Zahlenwert 0 zu tun und ist auch kein String wie "NULL".
- 2 Falle in MariaDB: "CONCAT() returns NULL if any argument is NULL."
- 3 Sie prüfen auf NULL nicht mittels = oder !=, sondern mit IS **NULL** oder IS **NOT NULL**.
- 4 Wenn Sie den Sonderfall NULL schreiben möchten, verwenden Sie keine Anführungszeichen. Sonst wird es für einen String gehalten.

## Sonderfall NULL-Werte II

Das können Sie mittels `IFNULL()` oder `COALESCE()` abfangen.

*Tipp für den Import von Daten:* Definieren Sie für das Feld einen Default-Wert, den zum Beispiel R, STATA oder SPSS als Fehlercode versteht. Wichtig ist dabei, dass Sie den Datentyp nicht wechseln und nicht die Zahl 0 verwenden.

## ORDER BY - geordnete Ausgabe

Der Befehl ORDER BY wird hinten an ein SELECT-Statement gehangen um die Ausgabe zu sortieren. Wichtig sind die Schlüsselwörter ASC (ascending / aufsteigend) und DESC (descending / absteigend). Beispiel:

```
SELECT firstName, age FROM persons ORDER BY  
age DESC ;
```

```
-----  
  firstName | age  
-----  
Methusalem | 110  
      Fritz |  80  
      Kevin |  26  
-----
```

# ORDER BY II

*Es kann nach mehreren Kriterien geordnet werden, welche durch Komma getrennt aufgeführt werden müssen.*

Beispiel:

```
SELECT firstName, lastName, age FROM persons  
ORDER BY age DESC, lastName ASC, firstName ASC;
```



# LIMIT I / II

Oft genügt es nur einen Teil der Daten zu sehen um einen ersten Eindruck zu bekommen. Es macht keinen Sinn von der Datenbank unter Umständen mehrere Millionen Zeilen anzufordern, wenn man nur an den ersten 10 interessiert ist.

LIMIT begrenzt die Zahl der zurückgegeben Zeilen. Sinnvollerweise ergänzt man das Statement um ein ORDER BY und WHERE Bedingungen.

## LIMIT II / II

So lassen sich dann zum Beispiel Ranglisten erzeugen:

```
SELECT name, citationCount  
FROM professors  
WHERE status = 'active'  
ORDER BY citationCount DESC  
LIMIT 3;
```

# Gute SELECT-Abfragen

## Eine gute SELECT-QUERY

- schränkt durch eine konkrete Auswahl (statt \*) die Zahl der Spalten ein,
- hat eine möglichst präzise WHERE-Bedingung um die Zeilenzahl zu reduzieren
- ordnet die Ausgabe nach einem sinnvollen Kriterium
- limitiert bei sehr großen Resultsets die Zahl der ausgegebenen Werte

# CONCAT I

Innerhalb eines SELECT-Statement verknüpft CONCAT() die Ausgabe verschiedener Spalten und / oder Strings miteinander.

**SELECT** CONCAT(expr1, expr2, expr3, expr4)

# CONCAT II

## geschachteltes CONCAT

In manchen Datenbanken kann CONCAT() lediglich zwei Elemente miteinander verbinden. Dann hilft es in der Regel

CONCAT-Befehle zu kapseln, also:

```
CONCAT(expr1, CONCAT(expr2, expr3))
```

Dieses Problem tritt zum Beispiel im MSSQL-Server 2012 innerhalb von Views auf.

AS

## AS I / III

Der Befehl `AS` erzeugt einen *temporären* Alias für eine Spalte oder eine Tabelle.

Das erlaubt uns innerhalb eines `SELECT`-Statement Spalten für die Ausgabe umzubenennen.

**SELECT** a, b, c **AS** d **FROM** table;

liefert die Spalten a, b und c der Tabelle, aber die Spalte c wird d benannt.

Besonders sinnvoll ist dieser Befehl, wenn wir `CONCAT` nutzen:

**SELECT** CONCAT(vorname, ' ', name) **AS** fullName **FROM** table;

## AS II / III

Das funktioniert ebenfalls mit Tabellen. Das ist gut für die Übersicht und spart Tipparbeit, wenn wir mit mehreren Tabellen arbeiten.

```
SELECT CONCAT(p.vorname, ' ', p.name) AS fullName, f.jobTitle  
FROM persons p LEFT JOIN functions f WHERE p.id = f.id;
```

statt

```
SELECT CONCAT(persons.vorname, ' ', persons.name) AS fullName,  
functions . jobTitle FROM persons LEFT JOIN functions  
WHERE persons.id = functions.id
```



## AS III / III

**Warnung: implizites AS**

Ein vergessenes Komma in der Liste der Columns wird von vielen DBMS (zum Beispiel MS SQLServer) implizit wie AS behandelt und erzeugt keine Warnung.

**SELECT** a, b c **FROM** table;

ist dann äquivalent zu

**SELECT** a, b **AS** c **FROM** table;

Oft wird eine solche Kurzschreibweise absichtlich benutzt:

**SELECT** p.name **FROM** persons p;

an Stelle von:

**SELECT** p.name **FROM** persons **AS** p;

# Berechnungen

# Berechnungen mit SELECT

Innerhalb Ihrer Abfrage können Sie bereits einfache Berechnungen durchführen. Etwa:

```
SELECT (revenue / 1000000) AS Mio FROM contractors;
```

# Aggregatfunktionen

Aggregatfunktionen sind im DBMS vordefiniert und oft sehr nützlich. Sie wenden die Funktion auf alle Werte der abgefragten Spalte an. Wenn das innerhalb eines SELECT Statement passiert, werden die Daten in der Tabelle nicht verändert, sondern nur die Ausgabe entsprechend angepasst.

Beispiele:

- Durchschnitt einer Spalte:  
**SELECT AVG(gehalt) FROM firma;**
- Werte zählen, die ein Kriterium erfüllen:  
**SELECT COUNT(\*) FROM <table> WHERE something = 1;**

# weitere Aggregatfunktionen

Üblicherweise unterstützt Ihr DBMS auch folgende Aggregatfunktionen:

- **MIN** : kleinster Wert in der Spalte
- **MAX** : höchster Wert in der Spalte
- **SUM** : Summe aller Werte in der Spalte

## Falle:

Etwas wie:

```
SELECT name, MAX(points) FROM persons;
```

wählt den ersten Namen in der Tabelle persons aus und den höchsten Punktwert aus der Gruppe aller Personen. Das Ergebnis stimmt nur, falls die erste Person *zufällig* auch den höchsten Punktestand hat!

Siehe: <https://mariadb.com/kb/en/library/max/>

# Eigene Funktionen / Stored Procedure

Sofern Ihr Account die nötigen Rechte innerhalb des DBMS hat, können Sie eigene Befehlssammlungen definieren und dauerhaft speichern.

Die meisten DBMS nennen eine solche Befehlssammlung Funktion, wenn Sie Werte liest und andere Werte zurück gibt. Wenn darüber hinaus Daten in Tabellen geändert werden, nennt man das meist Stored Procedure.

# Grenzen für Berechnungen in SQL

*SQL ist nicht für komplexe Berechnungen gemacht.* Einige DBMS erweitern die Zahl verfügbaren Funktion, aber für Regressionen, Zeitreihen, ... sollten Sie Statistik-Software wie zum Beispiel R benutzen.



## Tipps für R

R ist frei und Open Source. Es kann sich problemlos mit einem DBMS verbinden, wenn dessen Sicherheitseinstellungen diese Verbindung zulassen. Mehr dazu hier:

[www.ruediger-voigt.eu/accessing-databases-with-r.html](http://www.ruediger-voigt.eu/accessing-databases-with-r.html)

Allgemein:

- Als IDE für R ist RStudio zu empfehlen.
- Sie sollten eine aktuelle Anleitung nutzen, weil sich R schnell weiter entwickelt.
- Sehr praktisch sind die tidyverse-Packages und dabei insbesondere dplyr.
- Für Visualisierungen ist ggplot2 fast unschlagbar.

# Views

# Views

Wenn Sie eine komplexe SELECT-Abfrage geschrieben haben, dann müssen Sie diese nicht jedes Mal neu eingeben. Sie können Sie dauerhaft als View abspeichern.

# Tipps zum Vorgehen

- 1 Ermitteln Sie zunächst in welcher Tabelle / welchen Tabellen die gewünschten Daten zu finden sind.
- 2 Entwerfen Sie eine Abfrage, welche zunächst die benötigten Spalten ausgibt.
- 3 Schränken Sie das Ergebnis auf die nötigen Zeilen mit einer WHERE Bedingung ein.
- 4 Jetzt kümmern Sie sich um Berechnungen und Formatierung der Ausgabe.
- 5 Wenn die Abfrage gut funktioniert, können Sie sie als View abspeichern.

Weiter mit dem Foliensatz “Daten bearbeiten”.

Den kompletten Foliensatz finden Sie in aktueller Version unter:  
<https://www.ruediger-voigt.eu/kurs-datenbanken-und-sql.html>