Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

SQL1

Vorlesung Mai 2022

Letztes Update: 18. Mai 2022

Prof. Dr. Eva-Maria Iwer

Fachbereich Design Informatik Medien (DCSM)
Hochschule **RheinMain**

GLIEDERUNG

1. Einleitung
2. Die ersten Abfragen
3. Multirelations-Abfragen und Subabfragen

EINLEITUNG

EINLEITUNG

SQL

→ Der Zugriff auf eine RDMS erfolgt per SQL

→ Structured Query Language

→ Oft auch sequel ausgesprochen.

→ SQL ist eine High-Level Sprache

→ Sag was getant werden soll, aber nicht wie.

EINLEITUNG

SQL

SQL-Befehle lassen sich in vier Kategorien unterteilen:

→ (DQL) Befehle zur Abfrage und Aufbereitung der gesuchten Informationen

→ (DML) Befehle zur Datenmanipulation (Ändern, Einfügen, Löschen) und lesendem Zugriff

→ (DDL) Befehle zur Definition des Datenbankschemas

→ (DCL) Befehle für die Rechteverwaltung und Transaktionskontrolle.

Theme für heute

Befehle zur Abfrage und Aufbereitung der gesuchten Informationen

MARIADB

Wichtig

In dieser Vorlesung werden wir mariaDB 10.5.9 verwenden. Sie finden die Entwicklerdokumentation unter **https://mariadb.com/kb/en/sql-statements-structure/**).

Zugriff auf DB

Host: mariadb1 (nur von intern erreichbar, bzw. über VPN)
Service: mariadb
User-Name: (Informatik LDAP)
Password: (Informatik LDAP)
Database (wie user-name)
VPN: **https://doku.cs.hs-rm.de/doku.php?id=openvpn**

MARIADB

Allgemeines

- → Anzeige: MariaDB>
  - → Erwartet eine Eingabe
  - → MariaDB sendet diese zum Server zum Ausführen
  - → Anzeige des Resultats
  - → Ausgabe eines neuen MariaDB>
- → mehrzeilige Kommandos werden unterstützt
- → Ein Kommando wird normalerweise mit einem Semicolon ”;”beendet

MARIADB

Allgemeines

Anzeige	Bedeutung
MariaDB>	Erwartet neue Eingabe
->	neue Zeile bei mehrzeiligen Kommandos
'>	neue Zeile, erwartet das Schließen von einem String welches mit ' begann
" >	neue Zeile, erwartet das Schließen von einem String welches mit " begann
' >	neue Zeile, erwartet das Schließen von dem Kommando des Backticks
/*>	neue Zeile, erwartet das Schließen des Kommentars welche mit /* begann

MYSQL

Allgemeine Kommandos

Kommando	Bedeutung
h	Hilfe
q	Beenden
c	Kommando abbrechen
show databases;	alle verfügbaren Datenbanken anzeigen
Use [database_name];	Benutzen einer bestimmten Datenbank
select database();	Anzeige der aktuellen Datenbank
show tables;	Anzeige aller Tabellen in der aktuellen Datenbank
describe [table_name];	Anzeige der Tabellenstruktur

KOMMANDOS

Groß- und Kleinschreibung

Allgemeine Kommandokeyworte sind NICHT case sensitiv. Die Namen von Attributen und Tabellen sind es.

DIE ERSTEN ABFRAGEN

KOMMENTARE

→ Vom Zeichen # bis zum Zeilenende

→ Von der Sequenz −− bis zum Zeilenende

→ Von der Sequenz /∗ bis zur folgenden Seq. !*/!

NOTATION

```
SELECT
      [ALL | DISTINCT]
      select_expr [, select_expr ...]
     FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position}
                                [HAVING where_condition] ]
```

Wir verwenden folgenden Notation

→ Die in Großbuchstaben geschriebenen Wörter sind reservierte Bezeichner in SQL und sind bis auf Groß- und Kleinbuchstaben exakt so im Befehl anzugeben

→ Ausdrücke in eckigen Klammern sind wahlfrei. Die eckigen Klammern werden selbst nicht geschrieben!

NOTATION

→ Ausdrücke in geschweiften Klammern geben eine Auswahlliste an. Die einzelnen Listenelemente sind durch senkrechte Striche („|“) voneinander getrennt. Genau eine dieser Angaben ist auszuwählen. Die geschweifen Klammern und die senkrechten Striche werden nicht geschrieben!

→ Eine Auswahlliste kann statt mit geschweiften auch mit eckigen Klammern erfolgen. Dann ist maximal eine Angabe auszuwählen (keine ist auch eine Option!).

→ Drei Punkte „...“ weisen darauf hin, dass die vorherige Angabe beliebig oft wiederholt werden darf.

ANZEIGER ALLER DATEN

Einfachste Abfrage:

```
SELECT * FROM table_reference;
```

Liest alle Spalten einer Tabelle (* steht für alle)
Liest alle Zeilen einer Tabelle
Beispiel:

```
SELECT * FROM Drachen;
```

PROJEKTION

Auswahl der Spalten:

```
SELECT select_expr, ... FROM table_reference;
```

Nach SELECT können Spalten durch Komma getrennt angegeben werden

```
SELECT did, GivenName FROM Drachen;
```

Die Spalten können mit AS neu benannt werden

```
SELECT GivenName AS vorname FROM Drachen;
```

Mathematische Ausdrücke in der Select-Anweisung

```
SELECT did+1000 AS neueID FROM Drachen;
```

PROJEKTION

Konstanten in der Select-Anweisung

```
SELECT did, 'Jungdrache' AS Drachenalter
FROM Drachen
WHERE gj > 2000;
```

DELTA

Durch Verwendung von DISTINCT können doppelte Tupel ausgeschlossen werden
Beispiel:

```
SELECT DISTINCT (GivenName) FROM Drachen;
```

SORTIERUNG

Die Ausgabe kann folgendermaßen sortiert werden:

```
SELECT select_expr, ...
FROM table_reference
 [WHERE where_condition]
ORDER BY col_name [ASC|DESC][, ...];
```

Sortierung aufsteigend oder absteigend nach einer oder mehreren Spalten

SORTIEREN

Mit ORDER BY können die Sortierattribute angegeben werden. Beispiel:

```
SELECT * FROM Drachen ORDER BY GivenName, gj;
```

Die Reihenfolge kann durch ASC bzw. DESC angegeben werden.

```
SELECT * FROM Drachen ORDER BY GivenName ASC;
```

```
SELECT * FROM Drachen ORDER BY GivenName DESC;
```

```
SELECT * FROM Drachen
        ORDER BY GivenName DESC, did DESC;
```

SELEKTION

Bedingung, welche Daten ausgelesen werden:

```
SELECT select_expr, ...
       FROM table_reference
       WHERE where_condition;
```

OPERATOREN

nummerische Operatoren

Operator	Bedeutung
=	gleich
<>	ungleich
<	kleiner
>	größer
<=	kleiner gleich
>=	größer gleich

OPERATOREN

logische Operatoren

Operator	Bedeutung
AND	logische UND-Verknüpfung
OR	logische ODER-Verknüpfung
NOT	Negierung einer Bedingung

OPERATOREN

Operatoren bei Strings

Operator	Bedeutung
LIKE ”WERT”	suche nach einem bestimmten String (WERT)
LIKE “%WERT%”	suche nach einem String, der einen bestimmten WERT enthält. % kann dabei 0 bis n Zeichen sein
LIKE ”_WERT_”	suche nach einem String, der einen bestimmten WERT enthält. _ kann dabei genau 1 Zeichen sein
NOT LIKE ”WERT”	Negierung von LIKE ”WERT”
NOT LIKE “%WERT%”	Negierung von LIKE “%WERT%”
NOT LIKE ”_WERT_”	Negierung LIKE ”_WERT_”

SELEKTION - BEISPIELE

Alle Drachen die mit einem R beginnen

```
SELECT * from Drachen where GivenName like 'R%';
```

Alle Drachen die einen x im Namen haben

```
SELECT * from Drachen where GivenName like '%x%';
```

Alle Drachen die an der dritten Stelle im Namen ein x haben.

```
SELECT * from Drachen where GivenName like '__x%';
```

SELEKTION - BEISPIELE

Alle Drachen die vor 1800 geboren sind, männlich sind und der Name Christian ist. Die Ausgabe soll die did als DrachenID, GivenName als Name und gj als Geburtsjahr enthalten. Die Ausgabe soll nach Geburtsjahr absteigend sortiert werden.

```
SELECT did AS DrachenID,
       GivenName AS Name, gj AS Geburtsjahr
       FROM Dragon WHERE
            (gj < 1800 AND gs = 'male') AND
            GivenName='Christian'
                 ORDER BY gj DESC;
```

TUPEL MIT NULL-WERTEN VERHALTEN SICH BESONDERS

Betrachten wir ein Beispiel aus der Dragon Relation:

did	name	vater	mutter	gj	gs	art
1	Jessika	0	0	1650	NULL	4
2	Alexander	0	0	1680	male	3
3	Rene	0	0	1649	male	1

```
SELECT did, GivenName FROM Dragon
WHERE gs = "female" OR gs = "male";
```

Ergebnis:

did	name
2	Alexander
3	Rene

Ausgabe des Wahrheitswerts UNBEKANNT

Ergebnisse mit dem Wahrheitswert UNBEKANNT werden nicht angezeigt.

TUPEL MIT NULL-WERTEN VERHALTEN SICH BESONDERS

→ NULL wird niemals bei SUM, AVG oder COUNT dazuberechnet.

→ NULL kann niemals das MIN oder MAX sein.

TUPEL MIT NULL-WERTEN VERHALTEN SICH BESONDERS

Betrachten wir das Beispiel der Dragon Relation:

did	name	vater	mutter	gj	gs	art
1	Jessika	0	0	1650	NULL	4
2	Alexander	0	0	1680	male	3
3	Rene	0	0	1649	male	1

```
SELECT COUNT(*) FROM Dragon;
```

Ergebnis: 3

```
SELECT COUNT(gs) FROM Dragon;
```

Ergebnis: 2

AGGREGATOREN

Aggregatoren

Operator	Bedeutung
AVG	arithmetischer Mittelwert
MAX	höchster Wert
MIN	kleinster Wert
SUM	Summe
COUNT	Anzahl von Datensätzen (NULL wird nicht gezählt)

AGGREGATION

Das durchschnittliche Geburtsjahr aller Drachen:

```
SELECT AVG(gj)
FROM Drachen;
```

Das größte Geburtsjahr aller männlichen Drachen:

```
SELECT MAX(gj) as Jüngster
FROM Drachen
WHERE gs="male";
```

GRUPPIERUNG

Die Gruppierung erfolgt mit dem GROUP BY-Befehl:

```
SELECT select_expr, ...
FROM table_reference
 [WHERE where_condition]
GROUP BY col_name [, ...];
```

Wenn die Ergebnisgruppen noch eine bestimmte Bedingung erfüllen sollen, wird HAVING eingesetzt.

```
SELECT select_expr, ...
FROM table_reference
 [WHERE where_condition]
GROUP BY col_name [, ...]
HAVING where_condition;
```

GRUPPIERUNG

Das minimale Geburtsjahr aller Drachen per Art:

```
SELECT art, MIN(gj)
FROM Drachen GROUP BY art;
```

Drachenarten mit mehr als 15 Drachen die dazu gehören.

```
SELECT art, COUNT(did) AS Anzahl
        FROM Drachen
        GROUP BY art
        HAVING Anzahl > 15;
```

GRUPPIERUNG

Die ID und der Name vom ältesten Drachen per Art.

```
SELECT * FROM
        (SELECT art, MIN(gj) AS gj
        FROM Drachen GROUP BY art)t1
        NATURAL JOIN
        Drachen;
```

GRUPPIERUNG

Select Attribute in der Gruppierung

Wenn GROUP BY verwendet wird, dürfen die select_expr nur die Gruppierungsattribute und die Aggregationen enthalten.

FALSCH IST DAS FOLGENDE:

```
SELECT did, art, MIN(gj)
FROM Drachen group by art;
```

MULTIRELATIONS-ABFRAGEN UND SUBABFRAGEN

MULTIRELATIONALE ABFRAGEN

Allgemein

→ Die meisten Abfragen beruhen auf meherere Relationen

→ Wir können mehere Relationen in eine Abfrage integrieren, indem wir sie hinter FROM aufzählen (Kreuzprodukt)

→ Gleiche Attributsnamen können durch „<Relation>.<Attribute> “angesprochen werden.

→ Relationen können durch AS umbenannt werden bzw. kann das AS auch weggelassen werden

MULTIRELATIONALE ABFRAGEN

Formale Semantik

1. Gestartet wird mit dem Verbund der Relationen
2. Anwendung des Selektionsstatements (WHERE)
3. Projektion der Attribute und Ausdrücke (SELECT)

JOINS

Dokumentation

```
table_references:
    escaped_table_reference [, escaped_table_reference] ...

escaped_table_reference:
    table_reference
  | { OJ table_reference }

table_reference:
    table_factor
  | joined_table

table_factor:
    tbl_name [PARTITION (partition_names)]
        [[AS] alias] [index_hint_list]
  | table_subquery [AS] alias [(col_list)]
  | ( table_references )

joined_table:
    table_reference {[INNER | CROSS] JOIN | STRAIGHT_JOIN} table_factor [join_specification]
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_specification
  | table_reference NATURAL [INNER | {LEFT|RIGHT} [OUTER]] JOIN table_factor

join_specification:
    ON search_condition
  | USING (join_column_list)

join_column_list:
    column_name [, column_name] ...

index_hint_list:
    index_hint [, index_hint] ...

index_hint:
    USE {INDEX|KEY}
      [FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
  | {IGNORE|FORCE} {INDEX|KEY}
      [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

index_list:
    index_name [, index_name] ...
```

JOINS

Dokumentation

```
joined_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_specification]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON search_condition
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_specification
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

KREUZPRODUKT

Das Kreuzprodukt wird durch folgende Befehle hergestellt:

```
SELECT * FROM Drachen, Aufenthalt;
```

oder

```
SELECT * FROM Drachen CROSS JOIN Aufenthalt;
```

oder

```
SELECT * FROM Drachen JOIN Aufenthalt;
```

Ein Selbstjoin ist möglich durch:

```
SELECT distinct(d1.GivenName)
FROM Drachen d1, Drachen d2
WHERE d1.GivenName = d2.GivenName
AND d1.did > d2.did;
```

NATÜRLICHER VERBUND

Der natürliche Verbund wird durch folgenden Befehl hergestellt:

```
SELECT * FROM Drachen NATURAL JOIN Aufenthalt;
```

THETA-JOIN

Spezielle Form des Theta-Joins:

```
SELECT * FROM Drachen JOIN Aufenthalt
ON Drachen.did = Aufenthalt.did;
```

bzw.

```
SELECT * FROM Drachen JOIN Drachenkunde
ON Drachen.art = Drachenkunde.aid;
```

bzw.

```
SELECT * FROM Drachen INNER JOIN Drachenkunde
ON Drachen.art = Drachenkunde.aid;
```

OUTER-JOIN

In mariaDB 10.5 wird der Links- und Rechts-Outer-Join unterstützt. Dabei kann entweder der Natural-Join als Grundlage verwendet werden:

```
SELECT * FROM Aufenthalt
NATURAL RIGHT OUTER JOIN Drachen
ORDER BY did;
```

oder der Theta Join:

```
Select * From Aufenthalt
RIGHT OUTER JOIN Drachen
ON Aufenthalt.did=Drachen.did
ORDER BY Drachen.did asc;
```

Es gibt keinen Full-Outer-Join. Dazu muss der Rechts-Outer-Join und der Links-Outer-Join verbunden werden.

MULTIRELATIONALE ABFRAGEN

```
SELECT art, avg(gj)
FROM Drachen, Aufenthalt
WHERE Drachen.did > 15
GROUP BY art;
```

Reihenfolge der Abfragen

1. Als erstes wird der Join und die Select-Bedingung ausgeführt
2. Danach wird gruppiert
3. Dann die Projektion erzeugt

VEREINIGUNG, SCHNITT, DIFFERENZ

Vereinigung, ohne Duplikate

```
(SELECT did FROM Drachen)
UNION
(SELECT did FROM Aufenthalt);
```

Vereinigung, mit Duplikaten

```
(SELECT did FROM Drachen)
UNION ALL
(SELECT did FROM Aufenthalt);
```

VEREINIGUNG, SCHNITT, DIFFERENZ

Schnitt ohne Duplikate

```
(SELECT did FROM Drachen)
INTERSECT
(SELECT did FROM Aufenthalt);
```

Schnitt, mit Duplikaten

```
(SELECT did FROM Drachen)
INTERSECT ALL
(SELECT did FROM Aufenthalt);
```

VEREINIGUNG, SCHNITT, DIFFERENZ

Differenz, ohne Duplikate

```
(SELECT did FROM Drachen)
EXCEPT
(SELECT did FROM Aufenthalt);
```

Differenz, mit Duplikaten

```
(SELECT did FROM Drachen)
EXCEPT ALL
(SELECT did FROM Aufenthalt);
```

NULL WERTE FINDEN

```
SELECT * FROM
 Aufenthalt RIGHT OUTER JOIN Drachen
ON Aufenthalt.did=Drachen.did
WHERE Aufenthalt.did IS NULL
ORDER BY Drachen.did ASC;
```

SUBABFRAGEN

Allgemein

→ Eine umklammerte vollständige Abfrage (Select-From[-Where]) kann in einer Vielzahl von Plätzen verwendet werden (inkl. FROM und WHERE)

→ Beispiel: Anstatt einer Relation nach FROM können wir eine Subabfrage einfügen.

→ Wichtig ist nur, dass jede erstellte Relationen einen eindeutigen Identifier bekommt.

WEITERES BEISPIEL FÜR SUBABFRAGEN

Subabfrage in FROM und WHERE (Geben Sie den Namen und die DID von allen Drachen aus, welche jemals länger als die durchschnittliche Länge waren)

```
SELECT GivenName, did FROM Drachen NATURAL JOIN
        (SELECT did FROM Entwicklung WHERE
        laenge > (SELECT AVG(laenge) AS mittel
        FROM Entwicklung))t1;
```