
U8 – SQL

= Structured Query Language
(heute auch Standard Query Language)

ALLGEMEIN:

- Abfragesprache für relationale Datenbanken, die plattformübergreifend verwendet wird
- Vereinfachte Verwendung über Benutzeroberflächen wie phpMyAdmin
- In Verbindung mit der Skriptsprache PHP können die Abfrageergebnisse auf einer Website ausgegeben werden.

Befehle:

- Können groß oder klein geschrieben werden
- Werden mit ; abgeschlossen
- Reservierte Wörter (Anweisungen, Operatoren, Separatoren) dürfen nicht für Namen verwendet werden
- DB-Namen, Tabellennamen und Aliasnamen sind case sensitiv
- Aufbau: **Anweisung – Datenobjekte – Daten - Klauseln**

Anwendung:

- Auf Sonderzeichen in Feldnamen sollte verzichtet werden, auch wenn z. B. bei Access unter Verwendung von [] damit gearbeitet werden könnte
- Spaltenangaben bei Verwendung mehrerer Tabellen sollten mit *Tabelle .Spalte* angesprochen werden, auch wenn die Angabe der Tabelle obligatorisch wäre
- Dezimalpunkt statt Komma
- AUTO_INCREMENT-Felder enthalten immer einen eindeutigen Wert, da kein Wert doppelt vergeben wird. Wert wird auch nicht verändert, wenn vorherige Datensätze gelöscht werden
- Abfrage von Datumswerten variieren zwischen den einzelnen Datenbanksystemen
- Formeln werden in Bezug auf Datenbanksysteme als Ausdrücke betitelt und zwischen numerischen, alphanumerischen, logischen und Datumsausdrücken unterschieden

DATENTYPEN

- Feldtyp bestimmt, welche Daten gespeichert werden können
- Richtiger Typ ist für spätere Verwendung und die Systemperformance wichtig
- Typbestimmung beschränkt die Dateneingabe auf typspezifische Werte
- Numerischer Wert liegt vor, wenn mit ihm auch sinnvoll gerechnet wird, trifft z. B. bei Telefonnummer nicht zu
- Wenn Primärschlüssel Feldtyp AUTO_INCREMENT hat, muss der entsprechende Fremdschlüssel den Typ Integer haben
- Bei späterem Wechsel des Datentypes kann es zu Datenverlust kommen
- Lange Datenfelder verzögern Operationen wie Sortieren, Suchen, Filtern -> Bei großen Datenbanken deutlich längere Antwortzeiten

NUMERISCHE TYPEN

Smallint	kurze Ganzzahl	} Bereiche variieren je nach DBS
<u>Integer</u>	Ganzzahl	
Longint	lange Ganzzahl	
Float	Fließkommazahl mit einfacher Genauigkeit, bei manchen DBS auch Real bezeichnet	
<u>Double</u> (Precision)	Fließkommazahl mit doppelter Genauigkeit für wissenschaftliche Zahlen und Währungen	
Currency	Variante von Double, auf 2 Dezimalstellen gerundet und mit Währungszeichen versehen (in Access Währung)	
Decimal	Zahl mit Mindestvorgabe der Dezimalstellen	
Numeric	Zahl mit fester Vorgabe der Dezimalstellen NUMERIC (ANZAHL, NACHKOMMASTELLEN)	

- ↳ Decimal und Numeric mit exakter Darstellung benötigen mehr Platz als Double
- ↳ MySQL behandelt Numeric genauso wie Decimal

ALPHANUMERISCHE TYPEN

<u>Char</u>	Zeichenfolge fester Länge, in Access Text, je nach Datenbanksystem meist max. 255 Zeichen
<u>Varchar</u>	Variable Zeichenfolge, angegebener Wert setzt Maximallänge (auch Character Varying oder Char Varying)
Nchar	(auch National Char), Zeichenfolge für die Darstellung nationaler Zeichen, spezielles Charset (Zeichensatz) zugeordnet, variable Länge

- ↳ Varchar benötigt weniger Platz als Char, da Char mit Leerzeichen aufgefüllt wird, wenn der String zu kurz ist
- ↳ Für die Verwendung von Indizes ist Char jedoch besser geeignet

DATUMS- UND ZEITTYPEN

- Bei einigen DBS stellt der Unterschied nur eine Formatierung dar, Speicherung erfolgt trotzdem in Sekunden seit 1.1.1970 (= Unixtime)

Date	Datumsangabe YYYY-MM-DD
Time	Zeitangabe
Timestamp	Kombination aus Date und Time

SONSTIGE DATENTYPEN

Blob	(Binary Large Object), Speichert praktisch alle Arten von Daten als unstrukturierter Binärstrom, keine Interpretation der Daten, für Sound-Dateien, Videos, beliebige Grafikformate oder auch ausführbare Programme
Memo / Text	speichert lange Texte, je nach DBS reiner Ascii-Text und Beschränkung auf maximal 4000 Zeichen


OPERATOREN

MATHEMATISCHE OPERATOREN

+ - * / MOD

- MOD = Restwert einer Division (nicht bei allen DBS)
- Per SELECT direkt ausgeführt, Ergebnistabelle aus einzelнем Ergebniswert, um Bezeichnung zu wählen AS nötig -> virtuelle Spalte
- () zur Priorisierung
- Berechnung mit angegebenen und ausgelesenen Werten möglich
- Auch in WHERE-Klausel möglich

VERGLEICHSOPERATOREN

= < > <= >=  BETWEEN IN

- Kommt praktisch nur in WHERE-Klausel vor
- WHERE Spalte BETWEEN Wert1 AND Wert2
- BETWEEN auch für Datumswerte nutzbar
- IN erwartet Werteliste

LIKE NOT LIKE

- Zur Prüfung auf Teilidentität
- LIKE muss Suchmuster enthalten, NOT LIKE darf es nicht enthalten
- Suchmuster kann durch Ersatzzeichen % (beliebig viele Zeichen, * bei Access) und _ (genau ein Zeichen, ? bei Access) ergänzt werden
- Ersatzzeichen können beliebig oft im Suchmuster Vorkommen
- WHERE Spalte LIKE '%Suchmuster%'

LOGISCHE OPERATOREN

AND OR NOT

- AND = Beide Teilbedingungen müssen logisch wahr sein
 - OR = Mindestens eine Teilbedingungen muss logisch wahr sein
 - NOT = Kehrt den Wahrheitswert eines logischen Ausdrucks um
- WHERE NOT Spalte LIKE Wert

NULL

- Leerer Wert, nicht der numerische Wert 0
- Abfrage möglich mit IS NULL oder IS NOT NULL

FUNKTIONEN

→ Z. B. SELECT UPPER (Spalte) FROM Tabelle

- UPPER (Spalte) Ausgabe in Großbuchstaben (UCASE bei Access)
- INT Liefert ganzzahligen Anteil einer Dezimalzahl, Dezimalstellen werden abgeschnitten
- MOD Liefert Rest einer Division
- ROUND Rundet Wert auf eine Ganzzahl
- LOWER Ausgabe in Kleinbuchstaben
- TRIM Entfernt führende und folgende Leerzeichen
- COUNT Anzahl der Datensätze
- MAX höchster Wert einer Spalte
- MIN kleinster Wert einer Spalte
- AVG Durchschnittswert aller Einträge einer Spalte
- SUM Summe aller Werte einer Spalte

DATENBANKSTRUKTUR

DATENBANK ERSTELLEN, LÖSCHEN

Datenbank erzeugen

```
CREATE DATABASE Datenbank  
CREATE DATABASE IF NOT EXISTS Datenbank
```

- nicht in Access -> Erstellen über Abfrage nicht möglich
- Keine Leer- oder Sonderzeichen verwenden, _ ist zulässig
- Namen möglichst kurz halten

Datenbank löschen

```
DROP DATABASE Datenbank
```

→ Keine Sicherheitsabfrage, keine Möglichkeit rückgängig zu machen!

TABELLE ERSTELLEN, ÄNDERN, LÖSCHEN

Tabelle erzeugen

```
CREATE TABLE Tabelle (Spalte1 Typ(Länge), Spalte2 Typ(Länge),...)
```

Definierung Eigenschaften

```
Spalte Typ(Länge) Eigenschaft,...
```

PRIMARY KEY automatisch auch NOT NULL, UNIQUE
Nur unter MySQL mit obigem Befehl, in Access
..., PRIMARY KEY (Spalte),...

NOT NULL Darf kein leeres Datenfeld enthalten

UNIQUE Wert darf in Spalte nur einmal vorkommen

AUTO_INCREMENT automatisch auch UNIQUE
Kein Standardwert in SQL, Typ Integer, automatisch hochgezählter
Wert

DEFAULT Wird kein Wert übergeben, wird der Defaultwert eingetragen

Tabellenstruktur ändern

Spalte hinzufügen

- Begriff COLUMN optional

```
ALTER TABLE Tabelle ADD COLUMN Spalte Typ(Länge)
```

= als letzte Spalte einfügen

```
ALTER TABLE Tabelle ADD Spalte1 Typ(Länge) AFTER Spalte2
```

= Nach Spalte 2 einfügen

```
ALTER TABLE Tabelle ADD Spalte Typ(Länge) FIRST;
```

= An erster Stelle einfügen

Spalte ändern

```
ALTER TABLE Tabelle ALTER COLUMN Spalte Änderung
```

Spalte löschen

```
ALTER TABLE Tabelle DROP COLUMN Spalte
```

Tabelle löschen

```
DROP TABLE Tabelle
```

SELECT

- Zentraler SQL-Befehl
- Auch sehr komplexe Abfragen über mehrere Tabellen in wenigen Zeilen möglich

```
SELECT Spalte(n) FROM Tabelle
```

- ↳ Mehrere Spalten durch Komma getrennt
- ↳ Liefert alle Datensätze der Tabelle, zeigt aber nur die ausgewählten Spalten an
- ↳ Ausgabe der Spalten in der Reihenfolge, die in der Abfrage angegeben ist

```
SELECT * FROM Tabelle
```

- ↳ Liefert alle Datensätze und zeigt alle Spalten an
- ↳ Problematisch bei Memofeldern oder Feldern für Binärdaten wie Grafiken
- ↳ Ausgabe in der in der Tabelle angegebenen Reihenfolge

```
SELECT * FROM Tabelle WHERE Spaltenname='Wert'
```

- ↳ Gibt nur Datensätze aus, bei denen die WHERE-Bedingungen zutreffen
- ↳ Hochkomma um den Wert nur bei Zeichenfolgen (Strings) um Kollisionen zwischen Inhalt und Abfrage zu verhindern

```
SELECT DISTINCT Spalte FROM Tabelle
```

- ↳ Gibt keine doppelten Ergebnisse aus, Ausgabe weiterer Einträge wird unterdrückt, Einträge sind/bleiben trotzdem vorhanden

```
SELECT Spalte AS Aliasname FROM Tabelle
```

- ↳ Ergebnisse werden mit Aliasnamen statt Spaltennamen ausgegeben
- ↳ Relevant bei der Weiterverarbeitung in anderen Programmiersprachen

```
SELECT Spalte1 & ', ' & Spalte2 AS Aliasname
```

- ↳ Setzt mehrere Spalten zu einer zusammen (nur in Ergebnistabelle – gespeicherte Daten bleiben unberührt)
- ↳ Zusammengesetztes Ergebnis ist über Aliasnamen anzusprechen
- ↳ ', ' fügt zwischen die Spaltenergebnisse ein Komma mit Leerzeichen ein zur Trennung, beliebige alphanumerische Zeichen anwendbar

```
SELECT * FROM Tabelle ORDER BY Spalte
```

- ↳ Ergebnisliste wird nach der angegebenen Spalte sortiert A-Z
- ↳ Sortierung absteigend durch Anfügen von DESC
- ↳ Immer als letzte Angabe in der Abfrage **Spaltenliste -> Bedingung -> Reihenfolge**
- ↳ Mehrere Spalten zur Sortierung mit Komma getrennt, zweite Spalte wird verwendet, wenn in erster Spalte identische Werte enthalten sind
- ↳ Sortierrichtungen auch bei mehreren Spalten nutzbar
ORDER BY Spalte1 DESC, Spalte2 ASC

```
SELECT * FROM Tabelle ORDER BY Spalte LIMIT Zahl (MySQL)
```

```
SELECT TOP Zahl * FROM Tabelle ORDER BY Spalte (Access)
```

- ↳ Nur bestimmte Anzahl Datensätze wird ausgegeben
- ↳ Bei MySQL auch zwei Angaben möglich, um bestimmten Bereich auszugeben, Werte mit Komma getrennt

TABELLEN VERKNÜPFEN

- Als Verknüpfungsspalten geeignete Spalten nötig -> Schlüsselfelder

```
SELECT * FROM Tabelle1, Tabelle2
WHERE Tabelle1.Spalte1=Tabelle2.Spalte2
```

- Ohne WHERE-Klausel wird jeder Datensatz aus Tabelle1 mit jedem Datensatz aus Tabelle2 einzeln verknüpft

Join

Inner Join oder Natural Join

- Bildet Standard-Verknüpfung und stellt eigentlich eine Erweiterung der FROM-Klausel
- Benötigt ON-Klausel (wirkt wie WHERE-Klausel)

```
FROM Tabelle1 INNER JOIN Tabelle2
ON Tabelle1.Spalte=Tabelle2.Spalte
```

Variante 1: USING statt ON

```
USING (Spalte)
```

Setzt voraus, dass das Verknüpfungsfeld in beiden Tabellen die gleiche Bezeichnung trägt

Variante 2: FROM Tabelle1 NATURAL JOIN Tabelle2

Verknüpfungsspalten werden nicht angegeben, so dass genau eine Spalte in beiden Tabellen die gleiche Bezeichnung haben muss

- ↳ Varianten nicht in allen DBS
- ↳ Nur Datensätze, in denen die Einträge in den Verknüpfungsfeldern identisch sind werden angezeigt

Outer Join

- Alle Datensätze der linken bzw. rechten Tabelle werden angezeigt, auch wenn in der jeweils anderen Tabelle keine zugeordneten Datensätze enthalten sind
- LEFT OUTER JOIN zeigt alle Datensätze der linken (ersten) Tabelle, auch wenn in der rechten (zweiten) Tabelle dafür keine Datensätze enthalten sind
- Bezeichnung OUTER kann meistens entfallen
- Nicht vorhandene Daten in Tabelle2 werden mit Nullwerten angezeigt
- RIGHT JOIN sehr selten, nur in großen komplexen Datenbanken möglich, bzw. zum Aufspüren verwaister Datensätze
- Zusätzliche WHERE-Klausel möglich

Theta Join

- Liegt vor, wenn Verknüpfungsbedingungen kein Gleichheitsoperator ist, sondern Ungleich oder Größer-/Kleiner-Vergleich
- Eignen sich besonders dazu, in ähnlich strukturierten Tabellen bestimmte Beziehungen aufzudecken.

DATENSÄTZE ERSTELLEN, ÄNDERN, LÖSCHEN

Datensatz erstellen

```
INSERT INTO Tabelle (Spalte1, Spalte2,...) VALUES (Wert1, Wert2,...)
```

- Nicht angegebene Spalten/Werte bekommen NULL zugewiesen, wenn NULL nicht zulässig, wird der Datensatz nicht eingefügt
- Wenn alle Werte bei Values angegeben sind kann auf Angabe der Spalten verzichtet werden
- Datum mit YYYY-MM-DD angeben
- Berechnungen in Spaltenwerten zulässig

Datensatz ändern

```
UPDATE Tabelle SET Spalte=NeuerWert WHERE...
```

- Ohne WHERE-Klausel auf alle Datensätze angewendet
- Wert „löschen“ mit Änderung zu NULL

Datensatz löschen

```
DELETE FROM Tabelle WHERE ...
```

- Keine Rücknahmefunktion und keine Sicherheitsabfrage!
- Besser zuerst Markieren, z. B. durch zusätzliche Statusspalte; Abfragen werden dann erweitert um die Bedingung Status!=deleted