SQL-Übersicht

Structured
Query
Language

DB- und Tabellenstruktur

Datenbank

erzeugen	`CREATE DATABASE datenbank`
löschen	`DROP DATABASE datenbank`

Datentypen

Ganzzahlige Datentypen	
TINYINT	(1 Byte) Sehr kleiner Integer
SMALLINT	(2 Byte) kleiner Integer
MEDIUMINT	(3 Byte) mittelgroßer Integer
INTEGER (INT)	(4 Byte) Integer normaler Größe
BIGINT	(8 Byte) großer Integer
Fließkommazahl	
FLOAT	(4 Byte) Kleine Fließkommazahl
DOUBLE	(8 Byte) Große Fließkommazahl
DECIMAL	Gepackte „exakte“ Festkommazahl
Datentypen für Datum und Zeitangaben	
DATE	Datum. Format 'YYYY-MM-DD'
TIME	Zeitangabe. Format 'HH:MM:SS'
TIMESTAMP	Zeitstempel. (Aktuelle Systemzeit)
DATETIME	Kombination aus Datum und Uhrzeit.
YEAR	Jahr. Format 'YYYY'
Mengendatentypen	
ENUM	Definition einer Liste zulässiger Werte.
SET	Def. einer Menge zulässiger Werte.

Datentypen für Zeichenketten und Bytefolgen		
Zeichen	**Binär**	**Beschreibung**
CHAR	BINARY	Zeichenkette / Bytefolge
VARCHAR	VARBINARY	Zeichenkette / Bytefolge
TINYTEXT	TINYBLOB	Kurzer Text / Bytefolge
TEXT	BLOB	Normaler Text / Bytefolge
MEDIUMTEXT	MEDIUM-BLOB	Mittler Text / Bytefolge
LONGTEXT	LONGBLOB	Langer Text / Bytefolge

Tabelle

erzeugen	`CREATE TABLE tabelle ( merkmal1 datentyp1, merkmal2 datentyp2 )`

`UNSIGNED`	vorzeichenlose Zahl
`UNIQUE`	keine doppelten Werte möglich
`NOT NULL`	keine Nullwerte möglich
`AUTO_INCREMENT`	erzeugt eine fortlaufende automatische Durchnummerierung
`PRIMARY KEY`	definiert Tabellenschlüssel.
`SERIAL`	das Schlüsselwort ist ein Alias für: `BIGINT, UNSIGNED, NOT NULL AUTO_INCREMENT, UNIQUE`

umbenennen	`RENAME TABLE tabelle_alt TO tabelle_neu`
löschen	`DROP TABLE tabelle`

Tabellenspalte

hinzufügen	`ALTER TABLE tabelle ADD merkmal datentyp AFTER position`
modifizieren	`ALTER TABLE tabelle MODIFY merkmal datentyp`
umbenennen	`ALTER TABLE tabelle CHANGE merkmal_alt merkmal_neu datentyp`
löschen	`ALTER TABLE tabelle DROP merkmal`

Benutzerverwaltung

Benutzer

anlegen	`CREATE USER benutzer`
	`CREATE USER benutzer IDENTIFIED BY passwort`
Passwort einstellen	`SET PASSWORD FOR benutzer = PASSWORD(passwort)`
löschen	`DROP USER benutzer`

Zugriffsrechte erteilen

Es gibt verschiedene Rechte, die man auf unterschiedlichen Ebenen einem Datenbankbenutzer zuweisen kann. Die wichtigsten Rechte sind:

`INSERT`	Einfügerecht
`DELETE`	Löschrecht
`UPDATE`	Änderungsrecht
`SELECT`	Abfragerecht
`ALL PRIVILEGES`	Alle Rechte gleichzeitig

Ebenen

global	`GRANT RECHT ON *.* TO benutzer`
Datenbank	`GRANT RECHT ON datenbank.* TO benutzer`
Tabelle	`GRANT RECHT ON datenbank.tabelle TO benutzer`
Spalte	`GRANT RECHT (merkmal) ON datenbank.tabelle TO benutzer`

Rechte entziehen

Analog zum Erteilen der Rechte jedoch mit dem Befehl

```
REVOKE
```

Datenmanipulation

Daten

einfügen vollständig	`INSERT INTO tabelle VALUES(wert1, wert2, ...)`
einfügen teilweise	`INSERT INTO tabelle (merkmal1, merkmal2) VALUES(wert_1, wert_2)`
manipulieren	`UPDATE tabelle SET merkmal = wert WHERE bedingung;`
löschen	`DELETE FROM tabelle WHERE bedingung`

`NULL` Leere Referenz

Abfragen auf einer Tabelle

Bedingungen

Eine Bedingung

Die Bedingung ist ein Vergleich zwischen dem Merkmalswert und einem Vergleichswert. Mögliche Vergleichsoperatoren sind:

`=, <, >, <>, <=, >=`

```
SELECT merkmal_1, ..., merkmal_n
FROM   tabelle
WHERE  bedingung
```

Mehrere Bedingungen

Bei mehreren Bedingungen können diese mit folgenden logischen Operatoren verknüpft werden:

`AND, OR, NOT`

```
SELECT merkmal_1, ..., merkmal_n
FROM   tabelle
WHERE          bedingung_1
  VERKNÜPFUNG  bedingung_2 ...
```

Aggregationsfunktionen

Aggregationsfunktionen dienen dazu, aus den unterschiedlichen Merkmalswerten einer Ergebnismenge bzw. der Gruppe einen einzelnen Wert zu ermitteln.

Maximum	`MAX(merkmal)`
Minimum	`MIN(merkmal)`
Anzahl	`COUNT(*)`
Anzahl	`COUNT(merkmal)` *Anzahl ohne NULL-Werte*
Anzahl	`COUNT ( DISTINCT (merkmal) )` *Nur unterschiedliche Werte!*
Summe	`SUM(merkmal)`
Durchschnitt	`AVG(merkmal)`

Einfache Rechenoperationen

In SQL-Abfragen können einfache Rechnungen ausgeführt werden. Mögliche Rechenoperatoren sind:

`+, -, *, /`

Datumsfunktionen

Es werden unterschiedliche Datentypen für Datums- und Zeitangaben unterschieden. Die wichtigsten sind:

`TIME` (HH:MM:SS **Zeit**)
`DATE` (YYYY-MM-DD **Datum**)
`DATETIME` (YYYY-MM-DD HH:MM:SS **Datum + Zeit**)
`TIMESTAMP` (YYYY-MM-DD HH:MM:SS **Datum + Zeit**)

Datums-/ Zeitformate aus Einzelwerten

Datum	`MAKEDATE(jahr, tag_des_jahres)`
Zeit	`MAKETIME(std, min, sek)`

Datums-/ Zeitformate aus aktueller Systemzeit

Zeitstempel	`NOW()`
Datum	`CURDATE()`
Zeit	`CURTIME()`

Teilinformationen extrahieren

Monatsname	`MONTHNAME(datum)`
Wochtag-Name	`DAYNAME(datum)`
Wochtag-Nummer	`DAYOFWEEK(datum)`
Quartal	`QUARTER(datum)`
Kalenderwochennummer	`WEEKOFYEAR(datum)`

Teilinformationen extrahieren zu Zahlenwerten

Datum	`DATE(zeitstempel)`
Zeit	`TIME(zeitstempel)`
Jahr	`YEAR(datum)`
Monat	`MONTH(datum)`
Tag	`DAY(datum)`
Stunde	`HOUR(zeit)`
Minute	`MINUTE(zeit)`
Sekunde	`SECOND(zeit)`

Rechnen mit Datum und Zeit

Es können unterschiedliche Zeiteinheiten auf ein bestehendes Datum bzw. auf eine Zeitangabe addiert bzw. davon subtrahiert werden. Die wichtigsten Zeiteinheiten sind:

DAY, MONTH, YEAR, HOUR, MINUTE, SECOND

```
date + INTERVAL anzahl EINHEIT
```

Die Differenz zwischen zwei Datumsangaben in Tagen bzw. zwischen zwei Zeitangaben kann wie folgt ermittelt werden.

```
DATEDIFF(date1, date2)
TIMEDIFF(time1, time2)
```

Vergleichsfunktionen

LIKE

Als Bedingung im WHERE-Teil einer SQL-Anweisung kann ein Merkmal mit einem Suchmuster verglichen werden. Bei der Angabe des Suchmusters dürfen die folgenden Jokerzeichen verwendet werden:

% beliebig viele beliebige Zeichen ('*' bei MS-Access)
_ ein beliebiges Zeichen ('?' bei MS-Access)

```
... merkmal LIKE muster
```

BETWEEN

Diese Bedingung wird wahr, wenn der Wert des Merkmals zwischen dem angegebenen 'start'- und 'ende'- Wert liegt oder dem 'start'- bzw. 'ende'-Wert entspricht.

```
... merkmal BETWEEN start AND ende
```

IN

Diese Bedingung wird wahr, wenn der Wert des Merkmals einem der Werte 'wert 1' bis 'wert n' entspricht..

```
... merkmal IN (wert_1, ...wert_n)
```

Abfrage auf mehreren Tabellen

Equi-Join (Inner-Join)

Die Beziehung zwischen zwei Tabellen (und auch mehreren Tabellen) wird durch die Gleichsetzung des Schlüssel- / Fremdschlüsselpaares erreicht

```
SELECT  t1.merkmal_1, ..., t1.merkmal_n
        t2.merkmal_1, ..., t2.merkmal_n
FROM    tabelle1 t1, tabelle2 t2
WHERE   t1.schlüssel=t2.fremdschlüssel
```

```
SELECT  t1.merkmal_1, ..., t1.merkmal_n
        t2.merkmal_1, ..., t2.merkmal_n
FROM    tabelle1 t1 INNER JOIN
        tabelle2 t2
ON      t1.schlüssel=t2.fremdschlüssel
```

Left-Join / Right-Join

Wenn alle Datensätze einer Tabelle (Haupttabelle ht) angezeigt werden sollen, auch dann, wenn nicht zu jedem Datensatz ein zugehöriger Datensatz in der Verknüpfungstabelle (Nebentabelle nt) vorhanden ist, so muss mit den Schlüsselwörtern LEFT (links) bzw. RIGHT (rechts) die Position der Haupttabelle angegeben werden.

```
SELECT  ht.merkmal_1, ..., ht.merkmal_n
        nt.merkmal_1, ..., nt.merkmal_n
FROM    haupttabelle ht LEFT JOIN
        nebentabelle nt
ON      join-bedingung ht-nt
```

```
SELECT  ht.merkmal_1, ..., ht.merkmal_n
        zt.merkmal_1, ..., zt.merkmal_n
        nt.merkmal_1, ..., nt.merkmal_n
FROM    (  haupttabelle ht LEFT JOIN
           zwischentabelle zt
           ON  join-bedingung ht-zt   )
        LEFT JOIN nebentabelle nt
        ON join-bedingung zt-nt
```

Self-Join

Der Self-Join stellt eine Verknüpfung auf die eigene Tabelle dar und wird durch zwei Referenzen auf dieselbe Tabelle aufgelöst.

```
SELECT  t1.merkmal_1, ..., t1.merkmal_n
        t2.merkmal_1, ..., t2.merkmal_n
FROM    tabelle t1, tabelle t2
WHERE   join-bedingung t1-t2
```

Unterabfragen, INSERT-SELECT-Abfragen, VIEW

Unterabfragen (ein Rückgabewert)

Unterabfragen, die genau einen Rückgabewert als Ergebnis liefern, können direkt mit den folgenden Vergleichsoperatoren in einer Abfrage verwendet werden.

`=, <, >, <>, <=, >=`

Variablen

SQL-Variablen erlauben das Zwischenspeichern von Werten. Variablen können zur Programmierung von Funktionen und Prozeduren, aber auch zum Entzerren von verschachtelten Abfragen verwendet werden.

Variablennamen

Name	`@variable`
Wert-zuweisung	`@variable := wert`
	`SET @gehalt = wert`

Unterabfragen (Rückgabemenge)

Unterabfragen mit mehreren Rückgabewerten müssen mit den folgenden Mengenoperatoren in die Abfrage integriert werden.

ANY gilt für irgendein Element
ALL gilt für alle Element
IN Element in Menge enthalten
EXISTS Ergebnismenge existiert

INSERT-SELECT-Abfragen

Unterabfragen können im Zusammenhang mit der INSERT-Anweisung zum automatisierten Einfügen von Datensätzen verwendet werden.

VIEW

Ein View (Sicht oder auch Veränderliche genannt) ist eine, über einen eigenen Namen ansprechbare, virtuelle Tabelle, die auf einer Abfrage basiert.

erzeugen	`CREATE VIEW viewname AS ...`
modifizieren	`ALTER VIEW viewname AS`
anzeigen	`SHOW CREATE VIEW viewname`
löschen	`DROP VIEW viewname`

Sortieren, Gruppieren, Kombinieren

Sortieren

Am Ende einer Abfrage können mit dem Befehl ORDER BY ein oder mehrere Merkmale angegeben werden, nach denen die Ausgabe aufsteigend oder absteigend sortiert werden soll. Die Sortierreihenfolge wird mit den folgenden Schlüsselwörtern festgelegt

ASC (**asc**ending) aufsteigend
DESC (**desc**ending) absteigend

```
... ORDER BY merkmal1 ASC, merkmal2 DESC
```

Gruppieren

Mit der GROUP BY-Anweisung können ein oder mehrere Merkmale angegeben werden, gemäß derer die Ergebnismenge einer Abfrage in Gruppen aufgeteilt wird. Je Merkmal und Gruppe kann nur ein Wert ausgegeben werden. Daher müssen Merkmale, nach denen nicht gruppiert wird, in der SELECT-Anweisung mit Aggregationsfunktionen angegeben werden.

```
... GROUP BY merkmal1, ..., merkmal_n
```

Auf die Gruppen können mit der HAVING-Anweisung Gruppenbedingungen definiert werden, die von den Datensätzen einer Gruppe erfüllt werden müssen. Damit ergibt sich für SQL-Abgfragen der folgende prinzipielle Aufbau.

```
SELECT     anzeigemerkmale
FROM       quelltabellen
WHERE      einzelbedingungen
GROUP BY   gruppiermerkmale
HAVING     gruppenbedingungen
ORDER BY   sortiermerkmale
```

Kombinieren von Abfragen

Die Ergebnismengen unterschiedlicher Abfragen können zu einer einzigen Ergebnismenge mittels des folgenden Schlüsselwortes zusammengefasst werden, wenn Sie in der Anzahl und den Datentypen der Ausgabemerkmale übereinstimmen.

UNION Vereinigungsmenge

UNION-Abfragen

```
SELECT ...              1. SQL-Abfrage
UNION
SELECT ...              2. SQL-Abfrage
```

Anmerkung

Die Mengenoperationen INTERSECT (Durchschnitt) und EXCEPT (Differenz) sind im SQL-Sprachumfang von MySQL nicht enthalten, können aber durch einfache Abfragen mit IN oder EXISTS nachgebildet werden.

Trigger, Transaktionen und Indizes

Trigger

Ein Trigger ist ein über einen eigenen Namen ansprechbares Datenbankobjekt, das fest mit einer Tabelle verbunden ist. Es wird aktiviert, wenn für diese Tabelle ein bestimmtes Ereignis eintrifft.

erzeugen	`CREATE TRIGGER triggername`
	`zeitpunkt ereignis`
	`ON tabelle`
	`FOR EACH ROW`
	`  sql-anweisung`

- Der Zeitpunkt kann entweder **BEFORE** oder **AFTER** sein.
- Als Ereignisse werden unterschieden: **UPDATE**, **INSERT**, **DELETE**
- Zugriff auf bestehende oder neue Datensätze: **NEW**, **OLD**

löschen	`DROP TRIGGER triggername`

Transaktionsverwaltung

Mit Hilfe von Transaktionen können mehrere SQL-Befehle zu einer unteilbaren, atomaren Einheit zusammengefasst werden. Nach dem Start einer Transaktion können beliebig viele SQL-Befehle folgen. Es gibt abschließend zwei Möglichkeiten die Transaktion zu beenden. Entweder die Transaktion zurücknehmen oder das Transaktionsergebnis endgültig festschreiben. Eine Transaktion erfüllt die ACID-Eigenschaften:

- Atomicity atomar
- Consistency konsistent
- Isolation isoliert
- Durability dauerhaft

starten	`START TRANSACTION;`
festschreiben	`COMMIT;`
zurücknehmen	`ROLLBACK;`

Index

Durch das Anlegen von Indizes können die Zugriffszeiten für lesende Datenbankzugriffe auf bestimmte Merkmale deutlich verringert werden. Als Datenstruktur für den INDEX kann beim Erzeugen beispielsweise **BTREE** (B*-Baum) festgelegt werden.

erzeugen	`CREATE INDEX indexname`
	`  USING datenstruktur`
	`  ON tabelle(merkmal);`

anzeigen	`SHOW INDEX FROM tabelle;`
löschen	`DROP INDEX indexname` `ON tabelle`

Prozeduren und Funktionen

Mit Hilfe von gespeicherten Prozeduren und Funktionen können mehrere SQL-Anweisungen zu einer Einheit zusammengefasst werden, die dann über den Prozedur- bzw. den Funktionsnamen angesprochen werden kann. Im Gegensatz zu Prozeduren kann eine Funktion mit der RETURN-Anweisung einen Rückgabewert zurückliefern.

Prozeduren

erzeugen	`CREATE PROCEDURE`
aufrufen	`CALL`
verändern	`ALTER PROCEDURE`
löschen	`DROP PROCEDURE`

Funktionen

erzeugen	`CREATE FUNCTION`
aufrufen	`CALL`
Rückgabe	`RETURN`
verändern	`ALTER FUNCTION`
löschen	`DROP FUNCTION`

SQL im Internet

MySQL-Referenzhandbuch

Ein sehr gutes SQL-Referenzhandbuch (auch in deutscher Sprache) findet sich im Internet unter:

`http://www.mysql.com/`

Hinweis: Stellen Sie auf der Seite die Sprache auf 'Deutsch' ein und suchen Sie nach dem Begriff:
Referenzhandbuch