KNEBLER.DE

27.02.2011 - SQL-Referenz Mini

Datentypen in SQL

Die wichtigsten SQL-Datentypen sind

bool TRUE oder FALSE
int Ganzzahlen
char () String fester Länge
varchar () String variabler Länge mit festgesetztem Maximum
long varchar (unter einigen Datenbanken: Text mit mehr als 2048 Zeichen)
dec (,) Offensichtlich Festkomma. übersteigt ein Wert die Zahl der möglichen Stellen wird der höchstmögliche Wert gespeichert (bei dec(3,3) und 4321.4321 ist das z.B. 999.999).
float () Fließkommazahl
date Datumsangabe. Vorsicht: Datumsformate unterscheiden sich in den Datenbanken, MySQL möchte z.B. Daten in der Form 4 Zeichen für Jahr, 2 für Monat und 2 für Tag in der Formatierung YYYY-MM-DD. Außerdem werden Daten meist als Strings übergeben (also bei INSERT unbedingt auf Hochkomma achten). Angesichts all dieser Probleme und den Schwierigkeiten, z.B. herauszufinden wie viele Tage zwischen zwei DATEs lagen, beschränke ich meine Programmierung auf Unix-Timestamps: Die haben den Wert int.

SELECT

Einfaches SELECT
select ist der erste Schritt für eine Ausgabe aus einer Datenbank. Die Syntax ist SELECT FROM [WHERE **]. Es werden immer alle Zeilen einer Datenbank angesprochen, sind alle Spalten gemeint genügt ein *, ansonsten werden Spaltennamen (genau wie Tabellennamen) durch ein Komma getrennt.**
example.
select * from hauptbuchselektiert alle Zeilen mit allen Spalten aus hauptbuch.
Mit Funktionen, die die Programmiersprachen dann zur Verfügung stehen, können selektierte Einträge z.b. in Strings gespeichert werden (siehe: Eine kleine ODBC-Referenz).
select transaktion, datum from hauptbuchgewährt dann nur Zugriff auf die Spalten transaktion und datum (nur wenige Spalten auszuwählen vergrößert oft übersicht und verschnellert).

SELECT mit Bedingungen

select * from hauptbuch where (transaktion>40000)selektiert nur die Zeilen (alle Spalten, da *) einer Tabelle, bei denen Transaktion>40000 ist. Ein weiteres Beispiel ist
select kundennr, letztebestellung from faktura where (kundennr=192)

SELECT mit Sortierung

Syntax ist
order by [ascending/descending]select * from hauptbuch where (transaktion>40000) order by transaktion descendingselektiert z.b. bei einer Tabelle mit den Werten
wobei descending eine fallende Reihenfolge symbolisiert, ascending oder keine Angabe der Reihenfolge sortiert die Selektion in die andere Richtung.

JOINS

Oft werden die Einträge aus einer Tabelle in einer anderen Tabelle als Schlüsselwerte benutzt. Schlüsselwerte sind Werte, die in jeweils nur einer Zeile der Tabelle vorkommen. Z.B. wäre einen Kundendatenbank, in der jedem Kunden eine eindeutige Kundennummer zugeordnet ist, eine Datenbank mit Schlüsselwerten. Haben wir zwei Tabellen Können wir in SQL auch in einem Schritt nach einem Kundennamen suchen und den Zeitpunkt der letzten Bestellung ausgeben. Das nennt man dann einen JOIN aus beiden Tabellen.
Um gleichnamige Spaltennamen in verschiedenen Tabellen unterscheiden zu können schreibt man den Tabellennamen mit einem Punkt vor den Spaltenname:
select hauptbuch.transaktion, hauptbuch.datum from
hauptbuch
sieht zwar dumm aus, das ändert sich aber bei
select * from kunden, bestellungen where ((kunden.kundennr=bestellungen.kundennr) AND (kundenname='Fr. Brisbois?))
liefert Kundennr und Bestellzeitpunkt von Andreas Mühl zurück.
Die letzte anweisung ist für konventionelle Programmierer (zumindest ging es mir so) erst einmal schwer zu verdauen. Es zeit aber auch, das ein where kein if ist: where wird alle Spalten aller Datenbanken angewendet. So funktioniert die Geschichte auch wenn z.B. Anderas Mühl in der Kundendatenbank die erste Zeile innehat, in der Bestelldatenbank aber nur die 5te oder 1000ste.

CREATE

create ist für uns nur im Zusammenhang mit create table interessant. Es erzeugt

eine neue Tabelle in einer Datenbank. der Syntax ist
create table [,(...])z.B.:
create table kunden (kundennr int, name varchar(50))

INSERT

insert into < Tabellenname > values ([,...])Beachten Sie, dass der Wert sowohl bei Datumsangaben (i.d.r in der Form 'JAHR-MO-TG?) als auch bei Varchars (Strings) in Hochkomma (') eingeschlossen sein muss.
Eine Insert-Anweisung führt logischerweise zu Fehlern, falls nicht die gleiche Anzahl Werte in die Tabelle eingefügt werden soll, wie Spalten vorhanden sind.
Hin und wieder passiert es, dass ein Programmierer insert into tabelle1 value (2001, 'Jahr?) schreibt. Ein fehlendes s mach es dem Programmierer da bei der Fehlersuche schwer weil (nicht das korrekte values) value auch als Schlüsselwort erkannt wird.

DELETE

Zeilen löscht man in SQL mit
DELETE FROM WHERE < Bedingung >Beispiel:
DELETE FROM ANTIQUES WHERE ITEM = 'Ottoman?;

DROP

Drop ist das delete für ganze Tabellen. Mit
DROP löschen Sie die ganze Tabelle. Ich verwende es oft in den setup-Dateien, den Dateien, die Tabellen in den Datenbanken aufbauen. Vor dem Erstellen der Tabellen versuche man sie oft zu löschen, da ein Erstellen mit geänderten Spalten nicht möglich ist wenn die Tabelle noch in anderer Form existiert.

ALTER

Mit ALTER TABLE ADD/DROP COLUMN /kann man Tabellen im Nachhinein ändern ? praktisch.
Spalten hinzufügen
Spalten hinzufügen kann man z.B. mit
ALTER TABLE ANTIQUES ADD COLUMN (PRICE INT); Spalten löschen
... und Spalten löscht man mit ALTER....DROP:
ALTER TABLE ANTIQUES DROP COLUMN PRICE;

UPDATE

KNEBLER.DE

UPDATE SET = WHERE **ändert einen Eintrag in einer Datenbank (und ist als solches sehr praktisch in Administrationsseiten wo desöfteren in den Datenbankinhalten etwas geändert wird. Man denke nur an Preisänderungen in Online-Shops.)**
UPDATE ANTIQUES SET PRICE = 500.00 WHERE ITEM = 'Chair?;