# SQL Crash-Kurs

© Dr. Arno Schmidhauser
Letzte Revision: Januar 2005
Email: arno.schmidhauser@bfh.ch
Webseite: http://www.sws.bfh.ch/db

# Inhalt



Dieses Skript enthält vorgesehene Lücken für Ihre Kommentare.

# 1 Literatur


- "Relationale Datenbanken und SQL"; Günter Matthiessen, Michael Unterstein; Addison-Wesley, 3. Auflage, 2003.
- "SQL-3 Complete, Really"; Peter Gulutzan, Trudy Pelzer; Miller Freeman, 1999.
- "A Guide to the SQL Standard"; C.J. Date; Addison-Wesley 2000.


# 2 Was ist eine relationale Datenbank?

## 2.1 Logisch

- Eine relationale Datenbank ist eine Menge von Tabellen. Sowohl Nutzdaten wie Systemdaten sind in Tabellen abgelegt.
- Eine Tabelle ist im Wesentlichen definiert durch den Namen und den Datentyp der Spalten (Attribute, Kolonnen).
- Die Tabelle kann eine beliebige Menge von Datensätzen aufnehmen. Die Datensätze haben keine definierte Ordnung in der Tabelle.
- Es gibt nur atomare Datentypen!
- Tabellen sind mit Zugriffsrechten versehen.

## 2.2 Technisch

- Eine Datenbank ist ein Server-Programm, welches von seinen Client-Anwendungen SQL-Befehle entgegennimmt. Der Datenbank-Server läuft meist auf einem dafür vorgesehen Computer. Das Server-Programm kontrolliert und verwaltet die zu ihm gehörigen Dateien (Siehe Abbildung 1).
- Die Kommunikation zwischen Client und Server findet meist über eine TCP/IP Verbindung statt, über welche die Datenbank SQL-Befehle entgegennimmt und Abfrageresultate ausliefert.
- Der Server überwacht die Verbindung zu jedem Client und kann bei dessen Absturz einen Rollback unfertiger Arbeiten (Transaktionen) durchführen.

- Der Server steuert den Zugriff mehrerer Clients auf dieselben Daten, indem er während einer SQL-Operation die betroffenen Daten zuhanden *eines* Client sperrt.

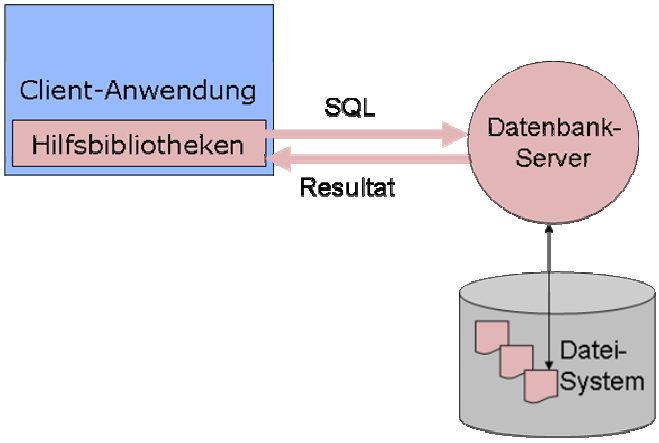


Abbildung 1: Datenbankserver

- In der Regel gehört zu einem applikatorisch sinnvollen Ablauf mehr als ein SQL-Befehl. Die Datenbank kann zusammengehörende SQL-Befehle zu Transaktionen bündeln. Eine Datenbank ist ein *Transaktionssystem*. Die Datenbank stellt sicher, dass eine laufende Transaktion im Fehlerfall rückgängig gemacht werden kann, dass die laufenden Änderungen für andere Clients erst nach Ablauf der Transaktion sichtbar sind, und dass die Änderungen nach Ablauf der Transaktion nicht mehr verlorengehen können.

## 3 Was ist SQL?

- SQL ist eine Sprache zum Verwalten, Bearbeiten und Abfragen von Tabellen in einer relationalen Datenbank.
- SQL heisst Structured Query Language und wurde von Oracle in den 70er Jahren erstmals in einem kommerziellen Produkt realisiert.
- Weil Daten in der Regel langlebig sind, kommt der kontinuierlichen Weiterentwicklung und Standardisierung von SQL grosse Bedeutung zu. Seit den 70er Jahren sind mehrere Generationen von

Programmiersprachen entstanden, während SQL sich als Sprache für den Datenzugriff global etabliert hat.

- Die SQL-Standards werden von ISO/ANSI herausgegeben, bisher in vier grösseren Schritten: 1989, 1992, 1999 und 2003. Das Grundwerk stammt von 1992 (häufig als SQL-2 oder SQL-92 bezeichnet.
- Im Gegensatz zu Programmiersprachen definiert man bei SQL nur *was* man tun will und nicht *wie* man es tun will. Der Algorithmus, also das *Wie*, wird bei SQL durch den Datenbankserver festgelegt. Folgendes Beispiel illustriert diesen Unterschied:

| SQL | C |
|---|---|
| **select** name<br>**from** Person<br>**where** id = '6' | f = **fopen** ("Person.dat", mode );<br>**while**(<br>fscanf(f, "%s%s ",id, name)> 0 ) {<br>**if**( strcmp(id, "6") == 0 ) {<br>printf( "%s %s", name );<br>}<br>}<br>**fclose** ( f ); |

Das SQL-Beispiel lässt die Art des Zugriffs offen, während das C-Beispiel eine sequentielle Suche definiert. Der Datenbankserver prüft meist Zugriffsmöglichkeiten und wählt darunter die schnellstmögliche aus: sequentielle Suche, logarithmische Suche auf sortierten oder indexierten Datensätzen.

- SQL ist keine kompilierte Sprache. Die Befehle werden erst unmittelbar beim Empfang im Datenbankserver verarbeitet:

  - Ist die Syntax korrekt?
  - Existieren die angesprochenen Tabellen und Attribute?
  - Sind die notwendigen Zugriffsrechte vorhanden?
  - Festlegung des Ausführungs-Algorithmus, eigentliche Ausführung und Auslieferung der Resultate.

Je nach Komplexität des auszuführenden Befehls beanspruchen diese Aufgaben einige Millisekunden bis gegen 100 Millisekunden, selbst wenn die Datenbank praktisch leer ist. Das führte fälschlicherweise zur Aussage, Datenbanken seien langsam. Zu

beachten ist aber, dass die Ausführungszeit meist nur sehr langsam *zunimmt*, selbst wenn der Datenbestand mehrere Gigabyte bis Terabyte umfasst. SQL-Befehle haben also hohe "Fixkosten", aber nur kleine "Variable Kosten".

## 4 Die 6 wichtigsten Befehle

Auch wenn SQL über 100 Befehle, mit teilweise mächtiger und komplexer Struktur, und mindestens noch einmal soviele Funktionen umfasst, kann doch das Wesentliche an ganz wenigen Befehlen erläutert werden. Im Folgenden sollen die SQL-Anweisungen

- create table
- drop table
- insert
- select
- update
- delete

etwas genauer angesehen werden.

Folgende Aussagen gelten generell für SQL-Befehle:

- Die Schlüsselwörter von SQL können gross oder klein geschrieben werden.
- Gross- und Kleinschreibung von Tabellennamen und Attributnamen wird nicht unterschieden. Beispielsweise ist die Tabelle `MESSWERT` identisch mit der Tabelle `MessWert`.
- Strings und Datumswerte werden in einfache Anführungszeichen gesetzt.
- Zeilenumbrüche werden wie Leerzeichen behandelt.
- Falls mehrere SQL-Befehle gleichzeitig abgesetzt werden, sind sie durch ein Strichpunkt ';' abzutrennen.
- Ob der Vergleich von Stringwerten sensitiv auf Gross-/Kleinschreibung ist, hängt von den Einstellungen in der Datenbank ab.

## 4.1 create table

Tabellen in SQL sind das zentrale Gefäss, um Daten aufzunehmen. Eine einmal erzeugte Tabelle verbleibt permanent in der Datenbank bis sie mit `drop table` wieder gelöscht wird. Vereinfachte Syntax:

```
create table tablename (
 attname typ [not null] [default wert] [constraint],
 …
)
```

In den nachfolgenden Beispielen wird eine Tabelle verwendet, welche Messwerte der Luftschadstoffe Ozon, $NO_2$ und $SO_2$ in der Region Bern enthält. Die Daten sind real gemessen worden im Juli 2003. Die in den Beispielen verwendeten Grenzwerte (120 yg und 80 yg) entsprechen den gesetzlichen Vorschriften. Beispiel für das Erstellen einer Tabelle:

```
create table Messwert (
  id      integer default autoincrement,
  station varchar( 20 ),
  zeit    timestamp,
  ozon    numeric(4,1) null,
  no2     numeric(4,1) null,
  so2     numeric(4,1) null
)
```

Wichtige Datentypen, die in praktisch allen Datenbank-Produkten vorkommen, und auch so heissen, sind `varchar(nnn)`, `date`, `time`, `timestamp`, `integer`, `real`, `double precision`, `numeric(s,d)` .
Der Datentyp `varchar(nnn)` ist für das Speichern von Zeichenstrings vorgesehen. Die Grösse von `nnn` ist produktspezifisch beschränkt, typischerweise auf `255`, `32'767` oder ähnlich.
Der Datentyp `numeric(s,d)` nimmt exakte Zahlen mit beliebiger Breite `s` und Dezimalstellen `d` auf. Die praktisch möglichen Werte von `s` und `d` sind produktabhängig, aber meist sehr gross, zum Beispiel `127`. Der intern benötigte Speicherplatz ist abhängig von `s` und `d`. `numeric` wird sehr häufig für Schlüsselwerte benutzt.

Die Datentypen `real` und `double precision` entsprechen den Typen `float` und `double` in Programmiersprachen, nehmen also eine nicht-exakte Zahl in intern 4 Byte respektive 8 Byte Speicherplatz auf[1].
Im Weiteren existieren Datentypen zur Aufnahme von unbeschränkt grossen Binär- und Textdaten. Im SQL-Standard heissen diese Datentypen `BLOB` und `CLOB`, in den konkreten Datenbank-Produkten aber häufig auch anders, beispielsweise `image` und `text`, oder `long binary` und `long varchar` usw.
Der create table Befehl kann viele weitere Angaben enthalten:

- Welche Attribute sollen eindeutige Werte enthalten
- Welche Attribute sind fakultativ
- Welche Attribute sind Schlüsselwerte
- Bedingungen, die der Wert eines Attributes erfüllen muss
- Defaultwert eines Attributes

## 4.2 drop table

Eine Tabelle samt Inhalt wird gelöscht mit dem Befehl

```
drop table tablename
```

Zu beachten ist der Unterschied zum Befehl
`delete from tablename`
Mit diesem Befehl wird nur der Inhalt der Tabelle, nicht aber deren Definition gelöscht.

## 4.3 insert

Datensätze werden mit dem Befehl

```
insert into tabellenname( spalte, ... )
values ( wert, ... )
```

in eine Tabelle eingefügt. Es kann immer nur *ein* Datensatz pro Befehl eingefügt werden. Jedoch können bei Bedarf mehrere ganze insert-Befehle

[1] Im SQL Standard ist die interne Speicherart (4 Byte oder 8 Byte) nicht festgelegt, wird aber in allen Datenbank-Produkten so gehandhabt.

in einem Zug an die Datenbank geschickt werden. Beispiel:

```
insert into Messwert ( station, zeit, ozon, no2, so2 )
values( 'Bern', '1.7.2003 18:00:00', 100.23, 8.6, 4.8 )
```

Beim Einfügen von Daten ist oft eine Konvertierung notwendig oder wünschenswert. Wird beispielsweie ein Zeitstempel von einer Applikation in Form von Sekunden seit dem 1.1.1970 bereitgestellt, wie das in Unix der Fall ist, so könnte der Einfügebefehl wie folgt aussehen (unter Zuhilfenahme der `dateadd()`[2]-Funktion von SQL):

```
insert into Messwert ( station, zeit, ozon, no2, so2 )
values( 'Bern', dateadd( second, wert, '1970-01-01' ),
        80.0, 20, 8.1 )
```

Um sehr viele Datensätze in eine Tabelle einzufügen, beispielsweise beim Initialisieren einer Datenbank bevor sie in den eigentlichen Betrieb übergeht, gibt es Befehle wie `load table ... from ...` . Diese sind jedoch produktspezifisch und müssen jeweils im Manual des Herstellers nachgeschlagen werden.
Wichtig ist, dass die Reihenfolge des Einfügens keinesfalls der Reihenfolge bei der Ausgabe der Datensätze entspricht. Der Datenbankserver ist frei, den Platz von vormals gelöschten Datensätzen für das Einfügen neuer Datensätze wieder zu verwenden. Die Ausgabe-Reienfolge ist nur definiert, wenn eine enstprechende Sortier-Angabe beim `select`-Befehl angegeben wird.

## 4.4 select

Der select-Befehl ist der mächtigste aller SQL-Befehle. Die Flexibilität und Geschwindigkeit, mit der eine Datenbank abgefragt werden kann, hat

[2] Die Funktion dateadd() ist an sich produktspezifsch. Der Standard sieht einen Datentyp INTERVAL vor für das Arbeiten mit Zeitintervallen. Leider kommt dieser Typ kaum in Produkten vor.

sicher den Erfolg von SQL wesentlich mitbegründet. Der select-Befehl hat im Wesentlichen die Struktur

| | |
|---|---|
| `select ...` | Auflistung von Attributen oder Ausdrücken mit Attributen und Funktionen |
| `from ...` | Auflistung der beteiligten Tabellen |
| `where ...` | Auswahlbedingung über die Datensätze |
| `group by ...` | Zusammenfassung der Datensätze in Gruppen |
| `having ...` | Auswahlbedingung über die Gruppe |
| `order by ...` | Attribute oder Ausdrücke für die Ausgabesortierung |

Ausführliche Beispiele:

```
select *
from Messwert
```

```
select station, zeit, ozon
from Messwert
```

```
select station, zeit, ozon
from Messwert
order by station, zeit
```

```
select zeit, so2
from Messwert
where station = 'Bern'
and zeit > '1.7.2003'
and zeit < '3.7.2003'
order by zeit
```

```
select station, zeit, ozon, no2
from Messwert
where ozon > 120
or no2 > 80
order by station, zeit
```

```
select distinct station
from Messwert
```

```
select count(*)
from Messwert
```

```
select min(ozon), max(ozon), avg(ozon)
from Messwert
```

```
select station, min(ozon), max(ozon), avg(ozon)
from Messwert
group by station
```

```
select station, min(ozon), max(ozon), avg(ozon)
from Messwert
group by station
having max(ozon) > 120
```

```
select datepart(day, zeit), min(ozon), max(ozon), avg(ozon)
from Messwert
group by datepart(day, zeit)
having max(ozon) > 120
order by datepart(day, zeit)
```

Obige Beispiele beziehen sich alle auf Abfragen mit *einer* Tabelle. Es ist aber eine wesentliche Stärke von SQL, dass mehrere Tabellen miteinander verknüpft werden können innherhalb einer Abfrage. Unsere Schadstoff-Verwaltungsdatenbank müsste in der Praxis aus mindestens 3 Tabellen bestehen:

- Eine Tabelle, in der die Stationsdaten abgelegt sind (der Standort beispielsweise)
- Eine Tabelle, in der die zu messenden Schadstoffe abgelegt sind (der Name und der Grenzwert beispielsweise)
- Eine Tabelle, in der die reinen Messdaten abgelegt sind (Zeit und Messwert beispielsweise). Diese Tabelle hat Verknüpfungen zu den beiden anderen, via Schlüsselwerte. Ein Abfragebeispiel möge hier zur Illustration genügen:

```
select Messung.zeit, Schadstoff.name, Messung.wert
from Station join Messung join Schadstoff
where Station.ort = 'Bern' and Schadstoff.name = 'ozon'
```

## 4.5 update

Mit `update` werden existierende Datensätze modifiziert. Es kann nur eine Tabelle gleichzeitig modifiziert werden.

| | |
|---|---|
| `update` *tablename*<br><br>`set` *attribut1* = *ausdruck1*,<br>*attribut2* = *ausdruck2*,<br>…<br>`where` … | Tabelle, in der Änderungen vorgenommen werden sollen.<br>Zuweisung eines neuen Wertes für eines oder mehrere Felder.<br><br>Auf welche Datensätze soll diese Änderung angewendet werden. |

Beispiel: Ozonwert um 5% korrigieren, weil die Station Bern falsch gemessen hat.

```
update Messwert
set ozon = ozon * 1.05
where station = 'Bern'
```

## 4.6 delete

Mit `delete` werden Datensätze aus Tabellen entfernt. Grundsätzliche Syntax

```
delete [from] tablename
where bedingung
```

Beispiel:

```
delete Messwert
where station = 'Bern'
```

# 5 SQL und Programmiersprachen

## 5.1 Interaktives SQL

- Für Administrationszwecke, Erstellen von Tabellen
- Für Aufräum- und Haushalt-Arbeiten, Datenpflege
- Für Zwischendurch-Abfragen, welche in fertigen Applikationsprogrammen nicht vorgesehen sind

## 5.2 SQL in Programmiersprachen

SQL kann in Programmiersprachen eingebettet werden. Hierzu gibt es umfassende Programmbibliotheken für kompilierte Sprachen wie C und Java, für Skriptsprachen wie ASP, PHP, VisualBasic usw.
Auch in Tabellenkalkulationsprogrammen wie Excel kann man SQL-Abfragen definieren, mit welchen dann Daten aus einer Datenbank abgezogen und beispielsweise grafisch dargestellt werden können.

Im Bereich C-Programmierung gibt es folgende Methodiken, um SQL zu verwenden:

- herstellerspezifische C-Bibliotheken
  - programmiertechnisch meist die einfachste Variante.
  - ergibt funktionssichere, schnelle Progamme.
  - Nachteil: herstellerspezifsch bereits im Source-Code.

- embedded SQL
  - Industriestandard, für viele Plattformen und Programmier-sprachen verfügbar.
  - Minimale herstellerspezifische Syntax. Klare Abgrenzung und Kennzeichnung der SQL-Aufrufe im Source-Code.
  - Die SQL-Befehle werden zur Kompilationszeit auf Korrektheit geprüft.
  - Nachteil: Precompiler notwendig, wirkt schwerfällig, arbeitet mit globalen Variablen.

- ODBC

- Open Database Connectivity, Datenbankschnittstelle von Microsoft. Gute Bibliotheken aber auch für Unix vorhanden, daher im Source-Code herstellerunabhängig.
- Sehr dynamisch: mehrere Verbindungen zu einer Datenbank gleichzeitig möglich, beliebige SQL-Befehle können zur Laufzeit an die Datenbank geschickt werden.
- Nachteil: Hohe Interaktion mit der Betriebssystem-Umgebung notwendig für die Definition von ODBC-Datenquellen. Etwas umständliche Programmierung. SQL-Befehle werden erst zur Laufzeit syntaktisch überprüft.
- In Web- und Desktop-Applikationen heute die bevorzugte Art der SQL-Anbindung für C-Programme. Das Pendant dazu in Java heisst JDBC (Java Database Connectivity).

Ein Beispiel für die Programmierung mit embedded C:

```
EXEC SQL INCLUDE SQLCA;

main( int argc, char* argv[] )
{
  /* Von SQL und C gemeinsam gebrauchte Variablen */
  EXEC SQL BEGIN DECLARE SECTION;
    char  station[20];
    float ozon;
    float no2;
    float so2;
  EXEC SQL END DECLARE SECTION;

  /* Hier den Variablen Werte zuweisen */
  /* ... */

  /* Verbindung aufbauen. */
  EXEC SQL CONNECT USING :verbindungsparameter;

  /* SQL-Befehl ausführen */
  EXEC SQL INSERT INTO Messwert( station, zeit, ozon, so2, no2 )
       VALUES ( :station, current timestamp, :ozon, :so2, :no2  );

  /* Aenderung bestaetigen */
  EXEC SQL COMMIT WORK;

  return( 0 );
```

```
}
```

Vollständiger Code in der Datei `Messwert.sqc`

# 6 Eigene Übungen

Für SQL-Versuche steht ein interaktiver Editor und ein Manual zur Verfügung unter:

www.sws.bfh.ch/db -> Dann zum Menupunkt 'Infos für Kurse und Klassen'