Structed Query Language (SQL)

Datenbankabfragen SQL

Für die Formulierung von Anfragen an eine Datenbank ist eine deklarative Sprache hilfreich
Deklarativ bedeutet, dass nur das WAS spezifiziert wird.
Das WIE (prozedural) wird dem Rechner überlassen.

Pseudocode ≙ prozedural ≙ WIE
SQL ≙ deklarativ ≙ WAS

Die (deklarative) Standartsprache für Datenbanken ist SQL.

Hauptbestandteile einer SQL Abfrage

SELECT (Startet eine Abfrage. Wie soll das Ergebnis aussehen?)
FROM (In welchen Tabellen stehen die Daten?)
WHERE (welche Bedingungen sollen berücksichtigt werden)
GROUP BY (Gruppierung von Zeilen)
HAVING (Gruppierungsbedingungen)
ORDER BY (Sortierung des Ergebnisses)
; ▷ Am Ende jeder Abfrage

Syntax:
SELECT Spaltenname
FROM tabellenname
{ WHERE Bedingung }

{ GROUP BY Spaltenname HAVING bedingung }
{ ORDER BY Spaltenname } ;

BSP.:

```
SELECT vorname, Name
FROM Mitarbeiter
WHERE Abteilung = 'controlling';
```

BSP 2: (bei mehreren Tabellen)

```
SELECT Kunde.KName
FROM Kunde, Rechnung
WHERE Rechnung.Datum = '2008-12-09'
AND Kunde.K-Nr = Rechnung.K-Nr;
```

Der Tabellenverbund (JOIN)

→ Informationen generieren unter der Nutzung mehrer Tabellen

Prinzipiell sind zwei Tabellen durch Kombination aller Zeilen zusammenführbar (Kartesischen Produkt)
Dies ist aber oft nicht gewünscht, es sollen nur passende Tupel kombiniert werden.
(einen Kunden sollen z.B. nur seine Rechnungen und nicht die Rechnungen aller Kunden zugeordnet werden)
Daher wird ein „Join" entlang der (Fremd-) Schlüssel der Tabellen (der Spalten, die beide Tabellen gemeinsam haben) durchgeführt

Doppelte Ergebniszeilen

Um in einer Abfrage doppelte Zeilen zu entfernen, verwendet man das Schlüsselwort DISTINCT direkt nach SELECT.

```
SELECT DISTINCT Kname
FROM ...
```

* Stern

- alle Spalten der Ergebnistabelle

```
SELECT *
FROM Kunde;        (wähle alle Spalten der Kundendatei aus)
```

Statische Funktionen

→ zur Verdichtung und Auswertung der abgefragten Daten

COUNT ( ) zählt Anzahl der Ergebniszeilen

z.B.

```
SELECT COUNT (*)
FROM Artikel       (Zähle alle Zeilen von Artikel)
```

SUM ( ) - bildet Summe des Klammerausdrucks
- nur bei numerischen Daten

z.B.

```
SELECT SUM (Gehalt)
FROM Personaldaten;
```

MIN / MAX ( ) • Sucht den minimalen / maximalen Wert

z.B.

```
SELECT MAX (Gehalt)
FROM Personaldaten;
```

AVG() • berechnet den Durchschnitt

z.B.

```
SELECT AVG(Gehalt)
FROM   Personaldaten;
```

Gruppierung:

GROUP BY tabellenname.Spaltenname

- Es werden diejenigen Zeilen zusammengefasst, die identische Attribute in der ausgewählten Spalte enthalten.
- Es ist darauf zu achten, dass andere Ergebnisspalten ebenfalls aggregierte Daten darstellen.

BSP.:

```
SELECT Geschlecht,
       AVG(Gehalt)
FROM Personaldaten
GROUP BY Geschlecht;
```

In der Select-Klausel muss das Attribut aus der GROUP BY-Klausel enthalten sein und die Werte müssen anhand anderer Attribute mittels Statistischer Funktion verdichtet werden.

BSP.:

```
SELECT
    Abteilung, COUNT(*)
FROM Mitarbeiter
GROUP BY Abteilung;
```

HAVING

Die HAVING-Klausel wird dem GROUP BY direkt nachgestellt und beinhaltet weitere Bedingungen für die Gruppierung (≙ der Wirkung des WHERE auf einzelne Zeilen.)

Bsp.:

```
SELECT Abteilung, SUM (Gehalt)
FROM Mitarbeiter
GROUP BY Abteilung
HAVING SUM (Gehalt) > 100.000;
```

Sortierung

ORDER BY [ASC | DESC] (aufsteigend / absteigend)

```
SELECT Name
FROM Mitarbeiter
ORDER BY Name ASC;
```

SQL 07

a)
```
SELECT Mitarbeiter.Personalnr, Mitarbeiter.Alter
  FROM Mitarbeiter, Kellner, bedient, Besucher
  WHERE Mitarbeiter.Personalnr = Kellner.Personalnr
    AND Kellner.Personalnr = bedient.Personalnr
    AND bedient.Kartennr = Besucher.Kartennr
    AND Besucher.VIP = 'ja'
    AND Kellner.Team = 'rot'
    AND Mitarbeiter.Alter > 25;
```

b)
```
SELECT Event-Nr, V-Umsatz
FROM Veranstaltung
WHERE Startdatum >= '01.03.2008'
AND Startdatum < '01.04.2008'
ORDER BY Umsatz DESC;
```

c)
```
SELECT Ranking, COUNT(*)
FROM veranstalter
GROUP BY Ranking;
```