Manipulations- & Abfragesprache für Datenbanken

1

SQL - Manipulations- & Abfragesprache für Datenbanken

Video zum Thema SQL (Grobüberblick)

Structured **Q**uery **L**anguage

Bezeichnet eine Sache zur Kommunikation mit Datenbanken.
Ist international genormt und wird von vielen DBS verstanden.

Eintragen, Ändern und Löschen von Daten (Data Definition Language - DDL)

=> Relationen definieren: CREATE
=> Primärschlüssel setzen mit: primary key
=> Tabellen löschen: DROP
=> Daten speichern: INSERT
=> Daten nachträglich verändern: UPDATE
=> Daten löschen: DELETE

Auswahl und Aggregation vorhandener Daten (Data Manipulation Language - DML)

=> SELECT
=> Selektion: WHERE
=> Ausgabe ordnen: ORDER BY
=> Join mit SELECT
=> Projektion

2

Es ist üblich, SQL-Schlüsselwörter GROSS zu schreiben, also CREATE statt create.

CREATE & Primärschlüssel mit primary key setzen

CREATE TABLE tabellenname (attribut1 datentyp primary key, attribut2 datentyp, attribut3 datentyp…);

CREATE TABLE lehrer (lehrernr integer primary key, Name varchar (20), Vorname varchar(20), Typ varchar(10), prnr integer);
CREATE TABLE schueler (schuelernr integer primary key, Name varchar (20), Vorname varchar(20), Klasse varchar(5), Geschlecht varchar(1));
CREATE TABLE projekt (prnr integer primary key, Thema varchar (30), Stufe varchar(15), Anzahl integer, Raum varchar(15), Geräte varchar(20));

Tabellen löschen: DROP

DROP TABLE tabellenname;

DROP TABLE sar_schueler;

3

Daten speichern: INSERT

INSERT INTO tabellenname (attribut1, attribut2, …. attributn) VALUES (wert1, wert2, …. wertn);

INSERT INTO lehrer (lehrernr, name, vorname, typ, prnr) VALUES (1,'Spiegel', 'Walter', 'Lehrer',1);
Es geht auch kürzer:
INSERT INTO lehrer VALUES (5, 'Hegel‘, 'G.‘ , 'Lehrer‘, 1);
Aber: Die Reihenfolge der einzelnen Daten eines Datensatzes muss streng eingehalten werden!

Daten nachträglich verändern: UPDATE

UPDATE tabellenname
SET attribut = attributwert
WHERE bedingung; ← *Wichtig sonst werden alle Datensätze geändert!!*

UPDATE lehrer
SET vorname = 'Georg'
WHERE lehrernr = 5;

Daten löschen: DELETE

DELETE FROM tabellenname
WHERE bedingung;

DELETE FROM lehrer
WHERE name = 'Hegel';

4

Abfragen mit SQL an Beispielen erklärt

Syntax einer (einfachen) SQL-Abfrage:

```
SELECT [Spalten]
FROM [Tabelle]
WHERE [Bedingung]
ORDER BY [Attribute];
GROUP BY [Attribute];
```

WHERE, ORDER BY und GROUP BY sind optional.

5

SELECT- Anweisung
Die SELECT-Anweisung ist eine sehr flexible Anweisung. Hier die einfachste Form:

```
SELECT   *
FROM     tabelle;
```

```
SELECT   *
FROM   schueler;
```

Der * ist ein Joker: es werden alle Spalten der Tabelle angezeigt.

Will man die Reihenfolge der Spalten bestimmen, so ruft man beispielsweise auf:
SELECT name, vorname, geschlecht, klasse, schuelernr
FROM schueler;

Datensätze mit Bedingungen auszuwählen - Selektion: WHERE

```
SELECT   spalte_1, spalte_2…
FROM     tabelle
WHERE    bedingung;
```

```
SELECT   name, vorname
FROM     lehrer
WHERE    name= 'Spiegel' AND NOT vorname= 'Heinz';
```

Bedingungen kann man durch logische Operatoren wie AND, OR oder NOT verbinden.

6

Auswahl aus mehreren Tabellen - Join mit SELECT

```
SELECT      attribut1, attribut2...
FROM        tabellenname1 INNER JOIN tabellenname2
ON          Bedingung;   ← meist ein Schlüsselfeldvergleich
```

```
SELECT      name, thema
FROM        lehrer INNER JOIN projekt
ON          lehrer.prnr = projekt.prnr;
```

Hier werden genau diejenigen Datensätze ausgewählt, deren prnr in beiden Tabellen identisch ist **(Join!).**

Damit die Ausgabe "schön" aussieht, können wir auch folgendes schreiben:

```
SELECT      name, thema
FROM        lehrer INNER JOIN projekt
ON          lehrer.prnr = projekt.prnr;
ORDER BY    name DESC;
```

Abkürzung für einen Tabellennamen - Alias: AS

Ein Alias wird wie im folgenden Beispiel benutzt :

```
SELECT      name, thema
FROM        lehrer AS L, projekt AS P
WHERE       L.prnr = P.prnr;
```

Abkürzung Abkürzung

7

Datenbankzugriff - Beispiele

```
SELECT   Name, Einwohner,Kontinent
FROM     Land
WHERE    Einwohner > 10;
```

Länder mit über 10 Mio. Einwohner

Land

LNR	Name	Einwohner	Hauptstadt	Kontinent
DK	Dänemark	5.16	Kopenhagen	Europa
D	Deutschland	81.34	Berlin	Europa
IND	Indien	761.00	Delhi	Asien
RWA	Rwanda	6.30	Kigali	Afrika
...	...	...	...	...

Ergebnistabelle

Name	Einwohner	Kontinent
Deutschland	81.34	Europa
Indien	761.00	Asien
...	...	...

9

Ein erstes Problem der Datenbank führt zur Erweiterung…
SQL – einfache Joins

Es sollen alle Länder mit ihren Kontinenten ausgegeben werden, die mehr als 10 Mio. Einwohner haben.

Land

LNR	Name	Einwohner	KNR
DK	Dänemark	5.16	EU
D	Deutschland	81.34	EU
IND	Indien	761.00	AS
RWA	Rwanda	6.30	AF

Kontinent

KNR	Name
EU	Europa
AS	Asien
AF	Afrika

Werden Daten aus mehreren Tabellen benötigt, werden Joins der Tabellen gebildet.

10

SQL – einfache Joins

1. Cross-Join - jede Zeile wird mit jeder kombiniert.

```
SELECT    *
FROM      Land INNER JOIN Kontinent;
```

LNR	Name	Einwohner	KNR
DK	Dänemark	5.16	EU
D	Deutschland	81.34	EU
IND	Indien	761.00	AS
RWA	Rwanda	6.30	AF

KNR	Name
EU	Europa
AS	Asien
AF	Afrika

LNR	Name	Einwohner	KNR	KNR	Name
DK	Dänemark	5.16	EU	EU	Europa
DK	Dänemark	5.16	EU	AS	Asien
DK	Dänemark	5.16	EU	AF	Afrika
D	Deutschland	81.34	EU	EU	Europa
D	Deutschland	81.34	EU	AS	Asien
D	Deutschland	81.34	EU	AF	Afrika
IND	Indien	761.00	AS	EU	Europa
IND	Indien	761.00	AS	AS	Asien
IND	Indien	761.00	AS	AF	Afrika
…	…	…	…	…	

DAS ERGEBNIS IST GROSSER UNSINN!

12

SQL – einfache Joins

2. Einschränken der Ergebnisse, es dürfen nur die Zeilen genommen werden, für die die Land und Kontinent-Tabelle Daten des gleichen Kontinents enthalten. Das wird durch die „Join-Bedingung" erreicht.

```
SELECT      Land.*, Kontinent.*
FROM        Land INNER JOIN Kontinent
ON          Land.KNR = Kontinent.KNR;
```

LNR	Name	Einwohner	KNR	KNR	Name
DK	Dänemark	5.16	**EU**	**EU**	Europa
~~DK~~	~~Dänemark~~	~~5.16~~	~~EU~~	~~AS~~	~~Asien~~
~~DK~~	~~Dänemark~~	~~5.16~~	~~EU~~	~~AF~~	~~Afrika~~
D	Deutschland	81.34	**EU**	**EU**	Europa
~~D~~	~~Deutschland~~	~~81.34~~	~~EU~~	~~AS~~	~~Asien~~
~~D~~	~~Deutschland~~	~~81.34~~	~~EU~~	~~AF~~	~~Afrika~~
~~IND~~	~~Indien~~	~~761.00~~	~~AS~~	~~EU~~	~~Europa~~
IND	Indien	761.00	**AS**	**AS**	Asien
~~IND~~	~~Indien~~	~~761.00~~	~~AS~~	~~AF~~	~~Afrika~~
…	…	…	…	…	

13

SQL – einfache Joins

3. Einschränken auf wichtige Spalten und umbenennen der Spalten.

```
SELECT      Land.Name AS Land, Land.Einwohner, Kontinent.Name AS KON
FROM        Land INNER JOIN Kontinent
ON          Land.KNR = Kontinent.KNR
WHERE       Land.Einwohner > 10;
```

LNR	Name	Einwohner	KNR	KNR	Name
D	Deutschland	81.34	EU	EU	Europa
IND	Indien	761.00	AS	AS	Asien
…	…	…	…	…	

Land	Einwohner	KON
Deutschland	81.34	Europa
Indien	761.00	Asien
…	…	…

14

SQL – Aliasnamen für Tabellennamen

In SQL werden lange Tabellenamen oft mit Alias-Namen abgekürzt.

Achtung das **AS** wird in **FROM** als Abkürzung und in **SELECT** als Umbennen gewertet.

Es sollen alle Städte mit mehr als 1 Mio. Einwohner ausgegeben werden; dabei auch das zugehörige Land.

```
SELECT    O.Name, L.Name,
FROM      Ort AS O INNER JOIN Land AS L
ON        O.LNR = L.LNR
WHERE     O.Einwohner > 10000000;
```

15