SQL,Teil 1:
CREATE, INSERT, UPDATE, DELETE, DROP

W. Spiegel

Übersicht

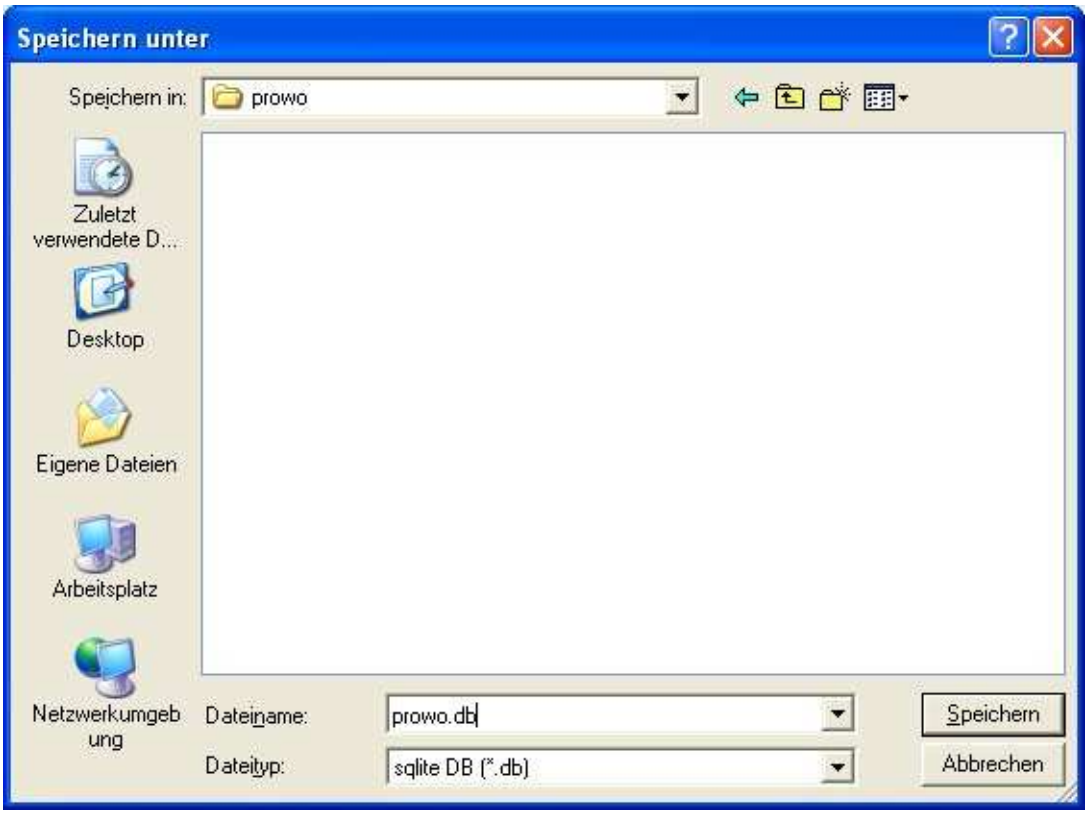
- DDL & DML
- Relationen definieren: CREATE
- Primärschlüssel setzen mit primary key
- Tabellen löschen: DROP
- Daten speichern: INSERT
- Daten nachträglich verändern: UPDATE
- Daten löschen: DELETE
- Beispiel
- Aufgaben

DDL & DML

DDL steht für Data Definition Language, das ist der Sprachteil von SQL, der zur Definition der Datenbank gedacht ist. Zur Datenmanipulation benutzt man den zweiten Teil der Befehle von SQL (deshalb **DML** : Data Manipulation Language), hierzu gehört vor allem die **SELECT** -Anweisung. Da wir erst mal eine Datenbank benötigen, fangen wir mit dem DDL-Teil an:

Relationen definieren: CREATE

Zuerst müssen wir in gfsqlite eine neue Datenbank ins Leben rufen: Gehe im Menü über Datenbank → Neue Datenbank erstellen, es öffnet sich folgendes Fenster:



Wurde die Datenbank erfolgreich erstellt, so antwortet gfsqlite:

und öffnet die Datenbank:

```
gfplus 1.1 -- Interaktive SQL shell
Monday November 17, 2003 08:59 PM
Benutze Datenbank "prowo.db"

pysql>
```

An dieser Stelle mein **Tipp** : benutze als Endung für sqlite-Datenbanken die Endung *.db, im Beispiel oben: prowo.db
Was jetzt noch fehlt sind Relationen (sprich: Tabellen), also vereinbaren wir welche mit CREATE:

```
pysql> CREATE TABLE sar_lehrer (lehrernr integer primary key,
       name varchar(20), vorname varchar(20), typ varchar(10),
       prowonr integer);
pysql> CREATE TABLE sar_schueler (schuelernr integer primary key,
       name varchar(20), vorname varchar(20), Klasse varchar(5),
       Geschlecht varchar(1));
```

Allgemein:

```
CREATE TABLE tabellenname (spalte1 typ, spalte2 typ, spalte3 typ);
```

Groß- oder Kleinschreibung kümmert gfsqlite nicht, es ist aber üblich, die SQL-Schlüsselwörter GROß zu schreiben, also CREATE statt create.

Mit dem Punkt Tabellen zeigen können wir unsere bisherige Arbeit kontrollieren:

PS: Das Fenster ist kontextsensitiv und liefert auch eine Übersicht der soeben vereinbarten Spalten!

Primärschlüssel setzen mit primary key

Wähle im Menü Datenbank den Punkt Struktur einer Tabelle zeigen und suche dir eine Tabelle aus, z. B. die Tabelle sar_projekt:

```
pysql> CREATE TABLE sar_projekt (prowonr integer primary key,
       Thema varchar(30), Stufe varchar(15), Anzahl integer,
       Raumbedarf varchar(15), SCHULGERAETE varchar(20))
```

Wichtig ist die Eigenschaft prowonr, unser Primärschlüssel! Wird jetzt eine Projektnummer doppelt vergeben, so zeigt gfsqlite eine Fehlermeldung im unteren Fenster an:

```
SQL Fehler bei der Ausführung des Befehls:

 INSERT INTO sar_projekt  . . .
```

Tabellen löschen: DROP

Mit dem Befehl DROP kann man Tabellen löschen, Beispiel:

```
pysql> DROP TABLE sar_schueler;
```

In gfsqlite gibt es **KEINE** Möglichkeit, die Struktur einer einmal definierten Tabelle im nachhinein zu verändern! Hat man also bei der Vereinbarung der Tabelle einen Väler (?!?) gemacht, so muss die Tabelle leider gelöscht werden.

VORSICHT: Nicht nur die Tabelle ist danach gelöscht, auch sämtliche Daten sind danach weg!!

Deshalb hier mein Ratschlag: speichere deine Sitzung über den Punkt
Datei → Sitzung speichern als . . .
als Textdatei ab (bitte die Endung *.txt nicht vergessen!)

Daten speichern: INSERT

Was wäre eine Tabelle ohne Daten? Also sorgen wir für etwas Information:

```
pysql> INSERT INTO sar_lehrer (lehrernr, name,vorname,typ,prowonr) VALUES
       (1,'Spiegel', 'Walter', 'Lehrer',1);
pysql> INSERT INTO sar_lehrer (lehrernr, name,vorname,typ,prowonr) VALUES
       (2,'Hilbert', 'David', 'Lehrer',2);
```

Statt der Langform geht es auch kürzer:

```
pysql> INSERT INTO sar_lehrer VALUES
       (5,'Hegel', 'G.', 'Lehrer',1);
```

Aber: Die Reihenfolge der einzelnen Daten eines Datensatzes muss streng eingehalten! Hier also zuerst die lehrernr, dann der Name, und so weiter.

PS: Einen ersten Überblick verschafft der Befehl

```
pysql> SELECT * FROM sar_lehrer;
```

allgemein:

```
pysql> SELECT * FROM <tabellen_name>;
```

Daten nachträglich verändern: UPDATE

Hat man sich bei der Eingabe vertan, oder muss man nachträglich Daten ändern, so benutzt man UPDATE:

```
pysql> UPDATE sar_lehrer
       SET vorname = 'Georg'
       WHERE lehrernr = 5;
```

Man kann damit in gfsqlite immer nur ein Attribut ändern, der folgende Befehl geht also "schief":

```
pysql> UPDATE sar_lehrer
       SET Typ = 'Schueler',
       SET Prowonr = 3
       WHERE lehrernr = 5;
```

Statt dessen müssen zwei UPDATE-Befehle eingegeben werden!

```
KORREKT:
pysql> UPDATE sar_lehrer
       SET Typ = 'Schueler'
       WHERE lehrernr = 5;

pysql> UPDATE sar_lehrer
       SET Prowonr = 3
       WHERE lehrernr = 5;
```

```
ERGEBNIS:
lehrernr | name      | vorname | typ      | prowonr
=====================================================
1        | Spiegel   | Walter  | Lehrer   | 1
2        | Hilbert   | David   | Lehrer   | 2
5        | Hegel     | Georg   | Schueler | 3
```

Daten löschen: DELETE

Mit DELETE kann man einzelne Datensätzte wieder löschen, Beispiel:

```
pysql> DELETE FROM sar_lehrer
       WHERE name = 'Hegel';
```

An dieser Stelle noch einmal der freundliche Hinweis:

Speichert eure Sitzungen ab, ihr spart wirklich Zeit!

Beispiel

Die Beispiel-Datenbank prowo.zip ins eigene Verzeichnis kopieren! Der Datenbanken-Reader ist diesmal leider keine Hilfe, schau dafür mal hier:

`http://www.highcroft.com/highcroft/sql_intro.pdf`
(DER SQL-Tutor, auf Englisch; wird demnächst lokal gespiegelt!),

ganz nett ist auch:

`http://www.oszhdl.be.schule.de/gymnasium/faecher/informatik/datenbanken/`

das Material zu SQL ist aber leider nur auf die SELECT-Anweisung begrenzt (den lernen wir das nächste Mal kennen)

Aufgaben

1. Kopiere die prowo-Datenbank prowo.zip in dein Verzeichnis (siehe oben)!
2. Erweitere die Datenbanken um einige Datensätzte: Projektleiter, Schüler, Projekte oder einfach ein paar Wahlen (INSERT). Anzeigen einer Tabelle geht so: **SELECT * FROM sar_lehrer;**
3. Jetzt wird es Zeit für die Korrektur der Datensätzte aus Aufgabe 2: wir sammlen Erfahrungen mit UPDATE.
4. Die Krönung: wir löschen Daten(sätze) (DELETE). Denke bitte an das Abspeichern von Sitzungen! gfsqlite kann Sitzungen importieren und "brav" Befehl für Befehl abarbeiten.
5. Meine Projektwahl ist voller Vähler: Zeit für das Löschen der Tabelle sar_wahl (DROP). Erstelle dann mit Hilfe von CREATE deine eigene Wahl-Tabelle sar_wahl und ergänze sie um ein einige Datensätzte. Ausgabe: **SELECT * FROM sar_wahl;**
6. Probiere auch die History-Funktion: <F5> Vorheriger Befehl, <F6> Nächster Befehl.
7. Wer zwischenzeitlich seine Sitzung abspeichert, hat mehr vom Leben!
8. Noch Zeit? Dann setze deinen Entwurf in Tabellen um.
9. Hausaufgabe: Mache dir ernsthafte Gedanken um die Modellierung deiner Miniwelt!

W. Spiegel, E-Mail: walter.spiegel@web.de