gfsqlite: ein einfacher SQL-Editor

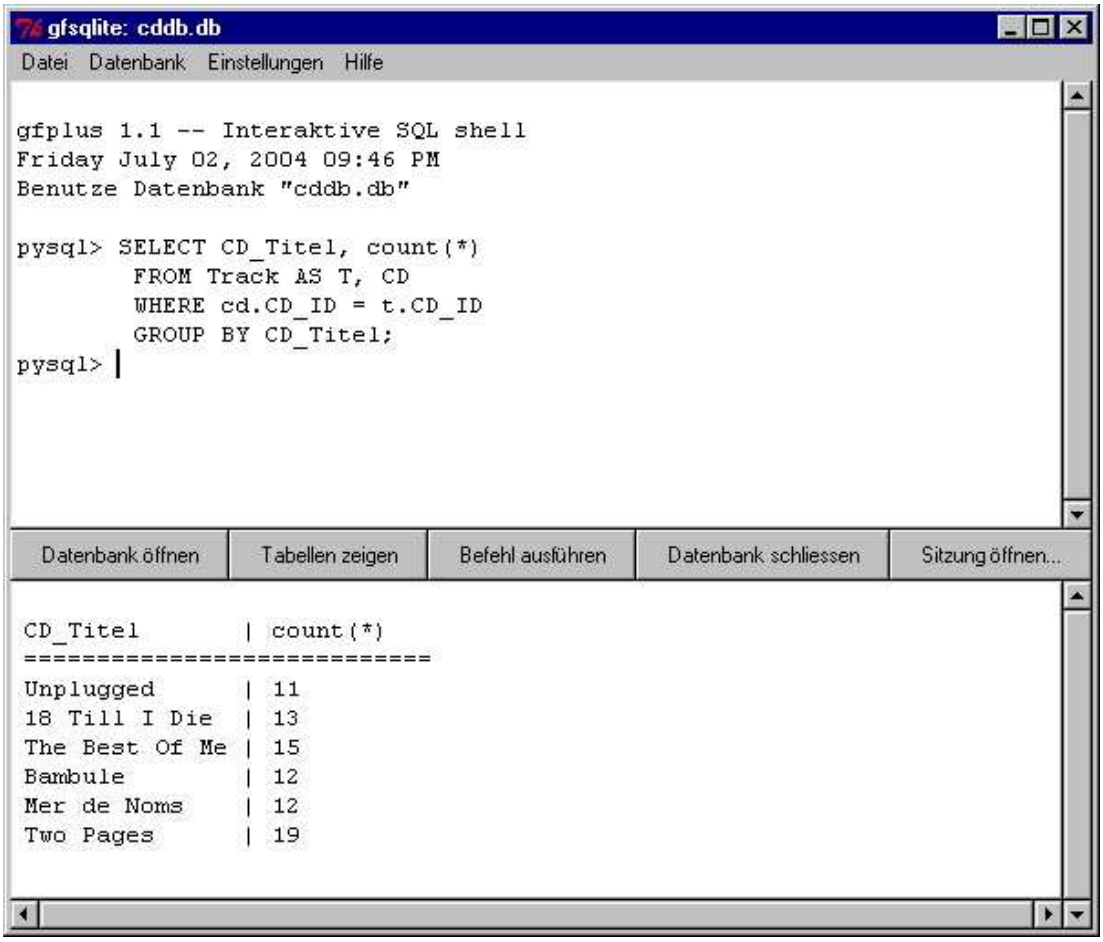
W. Spiegel

Übersicht

- gfsqlite
- Datenbank öffnen
- Buttons
- Menüs
- Einstellungen
- Arbeiten mit gfsqlite
- Crash-Kurs SQL
- Befehle & Tastenkombinationen
- Installation
- Links
- Credits

gfsqlite

gfsqlite beruht auf SQLite und der Datenbank-Schnittstelle pysqlite. SQLite ist eine relationale Datenbank, pysqlite erlaubt den Zugriff auf SQLite-Datenbanken unter Python. Eine Beispielsitzung mit der Datenbank **cddb.db**:

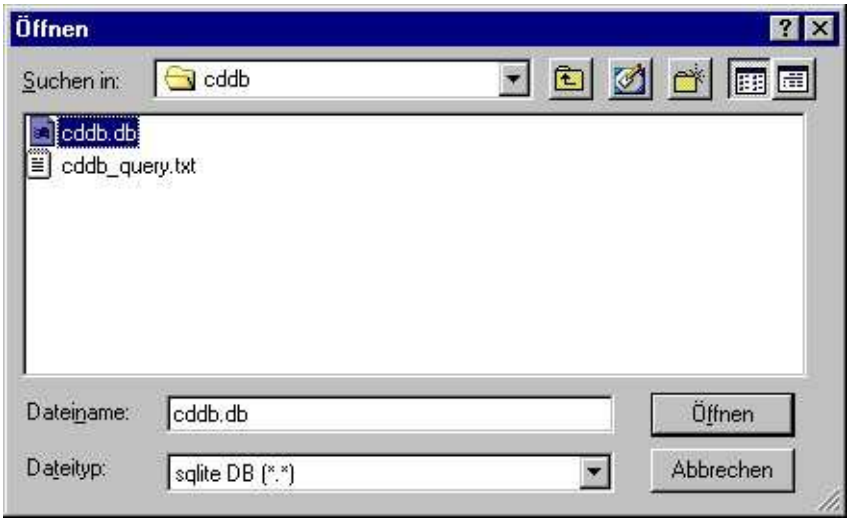


Im Fenster oben findet die Sitzung statt, hier werden SQL-Anweisungen eingegeben, `pysql>` ist der gfsqlite-Prompt. Eine SQL-Anweisung wird immer mit einem Strichpunkt **;** beendet, geht sie über mehrere Zeilen, so verändert sich der gfsqlite-Prompt zu `...>` , falls man zwischenzeitlich <**F4**> (=Befehl ausführen) gedrückt hat. Betätigen der RETURN- oder ENTER-Taste ergibt eine neue Zeile **ohne** den gfsqlite-Prompt `pysql>`. Im unteren Fenster werden Ergebnisse von Anfragen dargestellt: Das sind beispielsweise diejenigen Datensätze, die die SELECT-Anfrage erfüllen. Zusätzlich werden im unteren Fenster Fehlermeldungen oder alte Sitzungen angezeigt. In der blauen Titelzeile sieht man den Namen der momentan

geöffneten Datenbank. Für die Programmierung von gfsqlite wurde auf mehrere Tkinter-Routinen (tkSimpleDialog.py, tkDirectoryChooser.py) von Fredrik Lundh zurückgegriffen.

Datenbank öffnen

über den Button Datenbank öffnen , es öffnet sich folgendes Fenster:



SQLite-Datenbanken haben (in gfsqlite!) die Endung *.db (db = **D**aten**b**ank oder **d**ata**b**ase).

Buttons

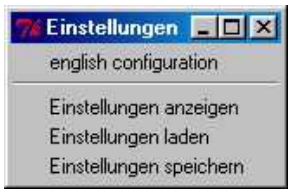
Datenbank öffnen	Öffnet eine Datenbank
Tabellen zeigen	Zeigt die Tabellen der momentan geöffneten Datenbank in einem neuen Fenster an. Beachte: das Fenster ist kontextsensitiv
Befehl ausführen	Führt den SQL-Befehl auf der momentan geöffneten Datenbank aus, alternativ: die Funktions-Taste <F4>
Datenbank schliessen	Schliesst die momentan geöffnete Datenbank, die Datenbank wird im aktuellen Zustand gespeichert!
Sitzung öffnen . . .	Öffnet eine alte Sitzung im unteren Fenster, Befehle können über Markieren, Kopieren &Einfügen in das obere Fenster übernommen werden.

Menüs

Einen Überblick über die Menüs gibt die folgende Seite.

Im Datei-Menü kann eine alte Sitzung geöffnet oder eine aktuelle Sitzung gespeichert werden. Über SQL-Datei einlesen kann eine Text-Datei mit SQL-Befehlen geöffnet und abgearbeitet werden. Es können auch Original-Sitzungs-Dateien geöffnet werden. Der bearbeitete SQL-Befehl wird im Fenster oben angezeigt. Bewirkt der SQL-Befehl eine Ausgabe, so wird sie im unteren Fenster dargestellt. Die beiden Befehle Fenster oben bzw. unten leeren schaffen Übersicht. Der Befehl DB speichern & schliessen beendet gfsqlite und speichert die momentan geöffnete Datenbank. Der Befehl Beenden beendet zwar gfsqlite, aber die momentan geöffnete Datenbank wird nicht gespeichert!

Neben Datenbank öffnen und schliessen, Befehl ausführen (alternativ: die Funktions-Taste <F4>), Tabellen anzeigen, Defintion aller Tabellen einer Datenbank anzeigen (Tabellen-Struktur zeigen) und Struktur einer Tabelle der momentan geöffneten Datenbank anzeigen kann auch eine neue Datenbank erstellt werden: Hierzu muss der Datenbank erst ein Name gegeben werden, danach kann man mit dem SQL-Befehl CREATE die Struktur der Tabellen (Relationen) eingeben. Der Punkt History öffnet ein Unter-Menü, in dem die letzten fünf geöffneten Datenbanken angezeigt werden. Über Lösche Befehlsliste wird der interne Puffer der SQL-Befehle (Tasten <F5>und <F6>) zurückgesetzt. Der letzte Punkt: "Exportiere Datenbank" bewirkt den Export der momentan geöffneten Datenbank als Textdatei.



In den Einstellungen kann auf die englische Konfiguration umgestellt werden. Daneben können die aktuellen Einstellungen angezeigt, geladen oder abgespeichert werden.

Das Hilfe-Menü

Daneben gibt es für das obere und das untere Fenster noch ein Popup-Menü: einfach im betreffenden Fenster auf die rechte Maustaste.

Einstellungen

Im Menü können über Einstellungen die aktuellen Einstellungen verändert werden, sie werden in der Datei "**gfsqlite.ini**" gespeichert:

Im Punkt "**Start-Sprache**" kann die Sprache eingestellt werden, in der gfsqlite starten soll. Bei "**locale**" wird versucht, die Systemeinstellung zu benutzen. Der Punkt "**Fenster löschen**" bewirkt beim Öffnen einer Datenbank das Leeren des oberen/unteren Fensters. Gleichzeitig wird auch die Historie der eingegebenen SQL-Befehle (Taste F5 bzw. F6) neu initialisiert, soll heißen: die ganze Geschichte beginnt von neuem . . .

Arbeiten mit gfsqlite

Aufruf: Doppelklick auf gfsqlite.pyw 

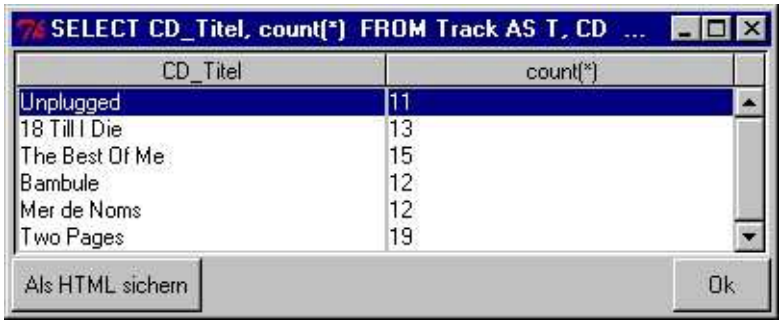
Alternativ: unter Windows `pythonw.exe gfsqlite.pyw`
unter Linux `python gfsqlite.pyw`
im gfsqlite-Verzeichnis aufrufen.
Vergleiche unter Linux auch die einfache shell-Datei `gfsqlite.sh` (Pfad anpassen!)

Das Arbeiten mit gfsqlite wird über **Markieren, Kopieren** und **Einfügen** wesentlich erleichtert! Deshalb sollte man unbedingt Sitzungen sichern, die Sitzungen werden als Textdateien gespeichert. Beim Öffnen dieser Textdateien wird der Inhalt im unteren Fenster von gfsqlite angezeigt. **Alternativ** kann man über den Befehl `SQL-Datei einlesen` auch eine Sitzungsdatei **vollständig** abarbeiten lassen, es werden hier jedoch **alle** SQL-Befehle der Sitzung ausgeführt!

Es gibt in gfsqlite eine **History-Funktion** : über <**F5**> kann man die vorherigen Befehle abrufen, über <**F6**> geht es in die andere Richtung. Man gebe einfach mehrere Befehle im oberen Fenster ein und probiere dann <**F5**> beziehungsweise <**F6**> .

Wichtig: Die über `SQL-Datei einlesen` verarbeiteten SQL-Befehle werden in die History-Liste aufgenommen, können also über die Tasten <**F5**> beziehungsweise <**F6**> abgerufen werden. Der Punkt History im Menü Datenbanken erlaubt dagegen Zugriff auf die letzten fünf geöffneten Datenbanken.

Mit der Tastenkombination <**F8**> kann die jeweils letzte SELECT-Anfrage als Pretty-Print-Tabelle dargestellt werden, Beispiel (gleiche Abfrage wie oben !):



Über `Als HTML sichern` kann man das Ergebnis der letzten SELECT-Anfrage auch als HTML-Datei abspeichern.

Zur Arbeit mit **commit & rollback** : gfsqlite speichert Daten nicht sofort physikalisch auf der Festplatte (!), sondern behält die Daten im Speicher! Man muss deshalb gfsqlite zum Abspeichern des aktuellen Zustandes explizit auffordern, hierzu dient der Befehl commit. Beispiel:

```
pysql> commit;
```

Beim Schliessen der Datenbank wird ebenfalls der aktuelle Zustand gespeichert, will man jedoch "zwischendrin" die Änderungen an der Datenbank rückgängig machen, so benutze man den Befehl rollback. Beispiel:

```
pysql> rollback;
```

Die Datenbank wird dann zurückgesetzt auf den letzten gespeicherten Zustand, technisch ausgedrückt: die Transaktion wird abgebrochen. Andererseits hat man mit dem Befehl commit die Möglichkeit, einen Zwischenzustand bei der Arbeit mit einer Datenbank physikalisch zu sichern.

Welche **Datentypen** stehen in gfsqlite zur Verfügung? Da gfsqlite auf SQLite aufbaut, wird das Thema Datentypen wie in SQLite behandelt. SQLite ist "typenlos", das bedeutet man kann in jeder Spalte einer Tabelle erst mal beliebige Daten speichern, egal was man vereinbart hat (!). Dazu die SQLite-Dokumentation: **"This behavior is a feature, not a bug"**. Intern speichert SQLite alle Daten als Text. Trotzdem ist es natürlich sinnvoll, bei der Vereinbarung der Tabellen einen Datentyp anzugeben, beispielsweise

- **VARCHAR(10)** für Zeichenketten (Strings), in Klammern wird die maximale Länge der Zeichenkette angegeben

- **INTEGER** für ganze Zahlen
- **FLOAT** für Kommazahlen
- **BOOLEAN** für boolesche Werte (wahr/falsch)
- **DATE** für das Datum

Der einzige "richtige" Datentyp in SQLite ist: **INTEGER PRIMARY KEY** , damit steht für den primären Schlüssel einer Tabelle eine AUTOINCREMENT-Funktion zur Verfügung. Hier eine Beispiel-Tabelle aus der CD-Datenbank:

```
CREATE TABLE CD (
  CD_ID INTEGER PRIMARY KEY,
  CD_TITEL VARCHAR(40),
  CD_JAHRGANG INTEGER,
  L_ID INTEGER,
  S_ID INTEGER,
  I_ID INTEGER
  );
```

Ansonsten ist der SQL-Standard relativ vollständig implementiert, näheres hierzu findet man in der Sprachbeschreibung von SQLite, beziehungsweise in der Dokumentation der (noch) nicht implementierten SQL-Eigenschaften.

Crash-Kurs SQL

Hinweis: die folgenden beiden Abschnitte befinden sich auch in der Hilfe-Datei von gfsqlite unter dem Menü-Punkt **Hilfe** .

Beispiele (`pysql>` ist der gfsqlite-Prompt):

- Erstellen einer Tabelle:

  ```
  pysql> CREATE TABLE mytest (id integer, inhalt varchar(20) );
  ```

- Datensatz einfügen:

  ```
  pysql> INSERT INTO mytest (id, inhalt) VALUES (1,'hallo');
  ```

- Spalten der Tabelle anzeigen lassen:

  ```
  pysql> DESC mytest;
  ```

- Eine Tabelle löschen:

```
pysql> DROP TABLE mytest;
```

- Datensatz löschen:

```
pysql> DELETE FROM mytest
       WHERE inhalt = 'hallo';
```

- Datensatz aktualisieren:

```
pysql> UPDATE mytest
       SET inhalt = 'pysqlsql'
       WHERE id = 1;
```

- Datensatz/Datensätze anzeigen:

```
pysql> SELECT * FROM mytest;
```

- Eine Spalte anzeigen:

```
pysql> SELECT inhalt FROM mytest;
```

- Datensatz/Datensätze sortiert anzeigen:

```
pysql> SELECT * FROM mytest
       ORDER BY id;
```

- Index auf das Attribut id setzen:

```
pysql> CREATE UNIQUE INDEX indexid
       ON mytest (id);
```

- View erstellen:

```
pysql> CREATE VIEW mini
       AS SELECT inhalt
       FROM mytest;
```

- View löschen:

```
pysql> DROP VIEW mini;
```

!Beachte den Strichpunkt ; am Ende eines Befehls!

Ansonsten vergleiche man den SQL-Tutor:
`http://www.highcroft.com/highcroft/sql_intro.pdf`

Befehle & Tastenkombinationen

Befehle in gfsqlite:

`<SQL Anweisung>;`	Eine SQL Anweisung in gfsqlite eingeben (mit ;beenden!)
`desc <relation>;`	Zeigt alle Spalten einer Relation an
`commit;`	aktuellen Zustand der Datenbank sichern
`rollback;`	Datenbank auf den letzten (!) gespeicherten Zustand zurücksetzen

Tastenkombinationen in gfsqlite:

`<F1>`	Anzeigen der Hilfe-Datei
`<F4>`	SQL Anweisung ausführen (alternativ: `<Alt>+X`)
`<F5>`	Vorherige SQL-Anweisung abrufen (alternativ: `<Alt>+P`)
`<F6>`	Nächste SQL-Anweisung abrufen (alternativ: `<Alt>+N`)
`<F8>`	Letzte SELECT-Anfrage als PrettyPrint-Tabelle (alternativ: `<Alt>+T`)
`<Alt>+C`	Fenster oben leeren

Installation

gfsqlite erfordert Python in der Version 2.2 aufwärts.

Die Installation von gfsqlite umfasst folgende Schritte:

1. Installation der Database-API pysqlite (Vers. 0.3.1 aufwärts) für den Zugriff auf SQLite. Hierzu muss gegebenenfalls der Quellcode von SQLite heruntergeladen werden (Sorry, dear Linux-User . . .), genaueres entnehme man der pysqlite-Dokumentation.

   ACHTUNG: Ohne die installierte Datenbank-Schnittstelle pysqlite wird gfsqlite nicht funktionieren!

2. Entpacken von **gfsqlite.zip** (Windows) bzw. von **gfsqlite.tar.gz** (Linux) in ein Unterverzeichnis. Bei der Installation wird im gfsqlite-Verzeichnis das Unterverzeichnis **gmodule** angelegt, in dem sich die gfsqlite-Module befinden.

3. Aufruf von gfsqlite.pyw im gfsqlite-Verzeichnis. Beim ersten Start von gfsqlite.pyw erzeugt der Python-Interpreter zu den Python-Skripten im gfsqlite-Verzeichnis einen Zwischencode, so dass der erste Aufruf *etwas* länger dauert. Danach sollte es *etwas* schneller gehen :-)

Während der Installation werden **keine** Veränderungen in der Registry vorgenommen (oder sonstwo . . .). gfsqlite wurde getestet unter Windows 98, Windows NT 4.0, Windows XP und Linux. Mir bleibt nur noch folgender

Hinweis: gfsqlite wurde vor allem zu didaktischen Zwecken geschrieben! Es ist nicht gedacht für die Entwicklung von großen & kritischen Datenbanken. Sondern: um SQL zu lernen. Have Fun!

Links

- Python: `http://www.python.org/`
- SQLite: `http://www.hwaci.com/sw/sqlite/`
- pysqlite: `http://pysqlite.sourceforge.net/`
- `MultiListbox.py`: `http://www.haucks.org/`
- gfplus: `http://endeavor.med.nyu.edu/~jeff/gfplus/`
- SQL-Tutor: `http://www.highcroft.com/highcroft/sql_intro.pdf`

Credits

Many Thanks to

- D. Richard Hipp for SQLite
- Michael Owens and Gerhard Häring for pysqlite
- Jeff Berliner for gfplus
- `http://www.haucks.org/` for MultiListbox.py (Codem Systems, Inc.)
- Fredrik Lundh for tkSimpleDialog.py & tkDirectoryChooser.py
- Guido van Rossum for Python

Hinweis: gfsqlite steht unter der BSD-Lizenz.

Feedback: `walter.spiegel@web.de`