Einführung in SQLite mit dem SQLite Manager von Firefox

Problem: Sie möchten für ihr Projekt Daten sammeln und später einer Gemeinschaft zur Verfügung stellen ohne proprietäre Formate zu verwenden oder servergebundene Datenbanksysteme vorauszusetzen.

Lösung: Sie verwenden SQLite, es ist u.a. für diesen Zweck entwickelt worden.

Problem 2: Sie arbeiten mit Windows und lange SQL-Anweisungen über mehrere Zeilen im Terminal sind nicht Ihre Stärke. Sie haben aber zumindest grundlegende Kenntnis von relationalen Datenbanken.

Lösung 2: Sie verwenden eine grafische Benutzeroberfläche (GUI, *grafik user interface*). Hier gibt es mehrere, der nachfolgend verwendete SQL Manager, ein add-on für Firefox ist nur eine Option.

Erwarten Sie bitte nicht, hier umfassen informiert zu werden, es gibt Bücher zu SQLite.

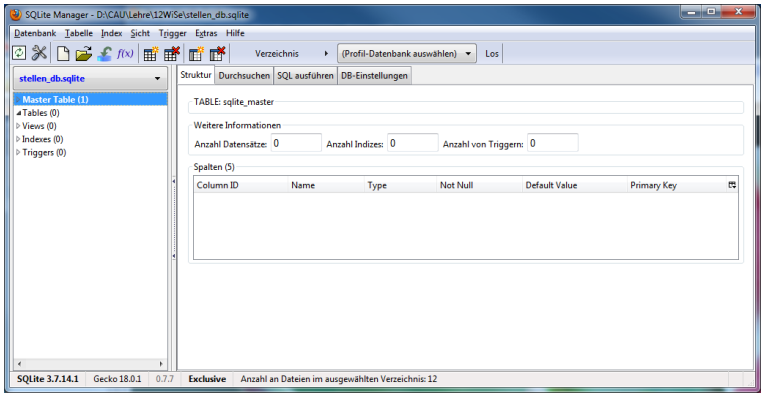

Abb. 1 SQLite Manager mit geöffneter stellen_db.sqlite.

1. Starten des SQLite Managers

Sie haben das add-on von Firefox bereits installiert und finden es im Menü des Browsers unter “Extras“.

Unter der Menüleiste folgt eine Iconleiste, an deren Ende Sie mit „Verzeichnis“ ein Standardverzeichnis wählen können, Vorgabe ist hier ihr Nutzerprofil von Firefox. Im folgenden *drop-down*-Feld können Sie die Datenbanken in diesem Standardverzeichnis auswählen. Das Programmfenster ist nachfolgend zweigeteilt.

Die linke Seite wird eingeleitet von einem weiteren drop-down-Feld, in dem Sie zwischen der aktiven Datenbank, den temporären und angehängten Datenbanken wechseln können. Darunter folgt die Liste der Tabellen, Sichten (gespeicherte Abfragen), Indices und Trigger (ereignisgesteuerte Prozeduren) der aktiven Datenbank.

Im rechten Fenster stehen vier Registerkarten zur Auswahl: Struktur, Durchsuchen, SQL ausführen, DB-Einstellungen. In Abhängigkeit von dem links markierten Objekt werden hier unterschiedliche Informationen angezeigt und weitere Optionen oder Aktionen angeboten.

2. Erstellen der Datenbank und der Tabellen

Wählen Sie im Menü “Datenbank“→“Neue Datenbank“, geben Sie der Datenbank einen Namen („stellen_db“) und wählen Sie anschließend ein Verzeichnis zum Speichern.

Prüfen Sie unter dem Reiter „DB-Einstellungen“ die korrekte Zeichencodierung (hier UTF-8).

Wählen Sie aus dem Menü “Tabelle“→“Tabelle erstellen“. Datenbank: „main“, also aktive Datenbank, Tabellenname: „tbl_DokFeldWert“, Attribute sind: DokFeldWertID (Integer, primary key autoincrement),

Stelle (Integer), Position (Integer), DokTyp (Text, Null erlauben), DokFeld (Text, Null erlauben), Wert (Text, Null erlauben), lastmoddate (datetime, current_timestamp).

- Jede Tabelle in SQLite erhält automatisch eine ROWID, es sei denn, Sie nennen eine Spalte ROWID oder erstellen eine Spalte vom Typ: integer primary key autoincrement (SQLite 3 autincrement).
- Ein Feld vom Typ timestamp ist immer eine gute Sache. Hier ist SQLite vermutlich etwas anders, als Sie es möglicherweise erwarten (SQLite3, Datentypen, Zeitfunktionen). Verwenden Sie: `CURRENT_TIMESTAMP`.
- Datentypen: SQLite speichert alles in den Datentype NULL, INTEGER, REAL, TEXT und BLOB. Die Unterscheidung in TEXT, CHAR und VARCHAR ist eher nur formalen Charakters, aber nicht bedeutungslos (SQLite3, Datentypen).

Erstellen Sie nun den zugehörigen Index: Markieren sie die Tabelle tbl_DokFeldWert, im Menü “Index“ → “Index erstellen“, Name: „index_tbl_DokFeldWert“, Doppelte Werte „nicht erlauben“, markieren Sie für folgende Attribute jeweils „Austeigend“: Stelle, Position, DokTyp, DokFeld.

In SQL das ganze knapp und bündig:

```
CREATE TABLE "tbl_DokFeldWert" ("DokFeldWertID" INTEGER PRIMARY KEY
AUTOINCREMENT, "Stelle" INTEGER NOT NULL , "Position" INTEGER NOT NULL ,
"DokTyp" TEXT, "DokFeld" TEXT, "Wert" TEXT, "lastmoddate" datetime current_timestamp,
CONSTRAINT "index_tbl_DokFeldWert" UNIQUE (Stelle, Position, DokTyp,
DokFeld));
```

Neue Tabelle erstellen. Name: „tbl_Positionen“, Attribute: Stelle (Integer), Position (Integer), Datum (Datetime, Null erlauben), Bearbeiter (Text, Null erlauben), DokTyp (Text, Null erlauben), Dokumentation (Text, Null erlauben), lastmoddate (Datetime, Null erlauben), createdate (Datetime, Null erlauben, current_date).

Das Ganze als SQL-Anweisung:

```
CREATE TABLE tbl_Positionen (Position_ID INTEGER PRIMARY KEY AUTOINCREMENT,
Stelle INTEGER NOT NULL , Position INTEGER NOT NULL , Datum DATETIME,
Bearbeiter TEXT, DokTyp TEXT, Dokumentation TEXT, lastmoddate DATETIME,
createdate DATETIME DEFAULT CURRENT_TIMESTAMP, CONSTRAINT
index_tbl_Positionen UNIQUE (Stelle, Position));
```

3. Mit den Daten arbeiten

Markieren Sie links die entsprechende Tabelle und wählen Sie rechts den Reiter „Durchsuchen“: der Inhalt der Tabelle wird angezeigt, ein Doppelklick auf einen Datensatz öffnet ein minimales Formular zur Dateneingabe, entsprechendes erfolgt auch mit den Schaltern „Datensatz hinzufügen“ am oberen Rand.

Unter dem Reiter „SQL ausführen“ können Sie SQL-Anweisungen eingeben und ausführen. Es gibt eine Reihe von Vorlagen für unterschiedliche Aufgaben, Standard ist „SELECT * FROM tablename“.

- Die SQL-Anweisung funktioniert auch ohne „;“ am Ende, gewöhnen Sie sich **nicht** daran!

4. Daten importieren

Mit „Datenbank“ → „Importieren“ wird der Reiter Importassistent ergänzt: CSV, SQL und XML mit zahlreichen Optionen sind möglich.

5. Daten exportieren

Die Ergebnisse von SQL-Abfragen können in CSV exportiert werden. Die Datenbank und alle Tabellen können mit „Datenbank“ → „Exportieren“ in SQL-Anweisungen exportiert werden (dump).

Autor: Christoph Rinne