Introduction to SQLite

Anthony Scemama <scemama@irsamc.ups-tlse.fr>

Labratoire de Chimie et Physique Quantiques
IRSAMC (Toulouse)

What is SQLite

- SQLite is an *an embedded relational database engine*.
- It allows to use SQL requests to put/get data into SQLite files.
- Very similar to PostgreSQL or MySQL, but there is no db server.
- Very useful to organize large amounts of data
- Can give some guarantees of consistence of the data

Available in most languages (Bash, Python, C, Java, Ocaml, Perl, Ruby, etc).

Introduction

Relational database

data organised in **Tables** with relations between tables

SQL

Computer language to manipulate data in a relational database

Table

Discribed by its **columns**. Each row is a **data item**

Primary key

Main column of a table, used to index the data. The primary key has to be *unique*.

Schema

Decription of the database in the SQL language

Elements table:

ID	Element
1	H
2	H
3	O

Coordinates table:

ID	x	y	z
1	0.0	0.7572	-0.4692
2	0.0	-0.7572	-0.4692
3	0.0	0.0	0.1170

Primary key is `ID` for both tables.

Running SQLite3

Run sqlite3 and create a `test.db` database file:

```
$ sqlite3 test.db
SQLite version 3.7.13 2012-06-11 02:05:22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

Help:

```
sqlite> .help
.backup ?DB? FILE      Backup DB (default "main") to FILE
.bail ON|OFF           Stop after hitting an error.  Default OFF
.databases             List names and files of attached databases
...
.timer ON|OFF          Turn the CPU timer measurement on or off
sqlite>
```

Schema

```
CREATE TABLE elements (
          Id    INTEGER PRIMARY KEY AUTOINCREMENT,
     Element    CHARACTER
);

CREATE TABLE coordinates (
          Id    INTEGER PRIMARY KEY,
           x    FLOAT,
           y    FLOAT,
           z    FLOAT
);
```

Adding data

```
INSERT INTO elements VALUES(1, 'H');
INSERT INTO elements VALUES(2, 'H');
INSERT INTO elements VALUES(3, 'O');

INSERT INTO coordinates VALUES(1, 0.,  0.7572, -0.4692);
INSERT INTO coordinates VALUES(2, 0., -0.7572, -0.4692);
INSERT INTO coordinates VALUES(3, 0.,  0.    ,  0.1170);
```

Showing data

Fetch a whole table:

```
sqlite> SELECT * FROM elements;
1|H
2|H
3|O
sqlite> SELECT * FROM coordinates;
1|0.0|0.7572|-0.4692
2|0.0|-0.7572|-0.4692
3|0.0|0.0|0.117
```

Select specific columns:

```
sqlite> SELECT y,z,id FROM coordinates;
0.7572|-0.4692|1
-0.7572|-0.4692|2
0.0|0.117|3
```

Select specific rows:

```
sqlite> SELECT * FROM elements WHERE element == 'H';
1|H
2|H

sqlite> SELECT * FROM coordinates WHERE z <= 0;
1|0.0|0.7572|-0.4692
2|0.0|-0.7572|-0.4692
```

Joining creates a new *view*, which is a temporary table containing the data of multiple tables.

`NATURAL JOIN` will join columns of with the same name.

```
sqlite3> SELECT * FROM elements NATURAL JOIN coordinates;
1|H|0.0|0.7572|-0.4692
2|H|0.0|-0.7572|-0.4692
3|O|0.0|0.0|0.117
```

Data can be printed in a specific order:

```
sqlite> SELECT * FROM coordinates ORDER BY y;
2|0.0|-0.7572|-0.4692
3|0.0|0.0|0.117
1|0.0|0.7572|-0.4692
```

Unique values can be printed:

```
sqlite3> SELECT DISTINCT z FROM coordinates ;
-0.4692
0.117
```

Deleting/Updating data

To delete data

```
sqlite3> DELETE FROM elements WHERE Id=2;
sqlite3> SELECT * FROM elements;
1|H
3|O
```

To update data:

```
sqlite3> UPDATE elements SET element='F' WHERE Id=3;
sqlite3> SELECT * FROM elements;
1|H
2|H
3|F

sqlite3> UPDATE elements SET element='O' WHERE element='H';
sqlite> SELECT * FROM elements;
```

```
1 | 0
2 | 0
3 | F
```

Operations

Print the coordinates table in atomic units:

```
sqlite> SELECT element, x/0.529177249, y/0.529177249, z/0.529177249
   ...>   FROM elements NATURAL JOIN coordinates;
H|0.0|1.43090051855196|-0.886659433841231
H|0.0|-1.43090051855196|-0.886659433841231
O|0.0|0.0|0.221097940663734

sqlite> SELECT sum(x), sum(y), sum(z) FROM coordinates;
0.0|0.0|-0.8214

sqlite> SELECT sum(x), sum(y), sum(z) FROM coordinates;
0.0|0.0|-0.8214

sqlite> SELECT element,x*x+y*y+z*z  FROM coordinates
  ... > NATURAL JOIN elements;
```

```
H|0.79350048
H|0.79350048
O|0.013689
```

Constraints

When a table is created, constraints can be set on columns:

NOT NULL

Ensure that a the value of the column is set for every row

UNIQUE

All row entries must be distinct

```
CREATE TABLE molecule (
    Mol_id  INTEGER PRIMARY KEY AUTOINCREMENT,
  Molecule  TEXT UNIQUE NOT NULL
);
```

```
sqlite> INSERT INTO  molecule(Mol_id,Molecule) VALUES(1,'CH4');
sqlite> SELECT * FROM molecule;
1|CH4
sqlite> INSERT INTO  molecule(Mol_id,Molecule) VALUES(2,'CH4');
Error: column Molecule is not unique
```

```
sqlite> INSERT INTO  molecule(Molecule) VALUES('H2O');
sqlite> SELECT * FROM molecule;
1|CH4
2|H2O
sqlite> INSERT INTO  molecule(Mol_id) VALUES(3);
Error: molecule.Molecule may not be NULL
```

FOREIGN KEY

Refers to the primary key of another table. Needs to be activated using `PRAGMA foreign_keys=1;`

```
CREATE TABLE method (
  Method_id  INTEGER PRIMARY KEY AUTOINCREMENT,
     Method  TEXT UNIQUE NOT NULL
);

CREATE TABLE computation (
  Computation_id  INTEGER PRIMARY KEY AUTOINCREMENT,
        Method_id  INTEGER NOT NULL,
```

```
          Mol_id  INTEGER NOT NULL,
  FOREIGN KEY(Method_id) REFERENCES method(Method_id),
  FOREIGN KEY(Mol_id) REFERENCES molecule(Mol_id)
);
```

```
sqlite> INSERT INTO method(Method) VALUES("Hartree-Fock");
sqlite> INSERT INTO method(Method) VALUES("MP2");
sqlite> INSERT INTO method(Method) VALUES("QMC");
sqlite> INSERT INTO method(Method) VALUES("CCSD(T)");

sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(1,1);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(1,2);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(2,2);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(3,2);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(3,1);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(4,2);

sqlite> SELECT * from computation;
```

```
1|1|1
2|1|2
3|2|2
4|3|2
5|3|1
6|4|2

sqlite> SELECT Method,Formula from computation
   ... >  NATURAL JOIN method
   ... >  NATURAL JOIN molecule;
Hartree-Fock|CH4
Hartree-Fock|H2O
MP2|H2O
QMC|H2O
QMC|CH4
CCSD(T)|H2O

sqlite> DELETE FROM method WHERE Method = "MP2";
```

```
Error: foreign key constraint failed
```

CHECK

Checks a constraint when the data in inserted

```
CREATE TABLE energy (
                id  INTEGER PRIMARY KEY AUTOINCREMENT,
  Computation_id  INTEGER NOT NULL,
          Energy  REAL NOT NULL CHECK(Energy < 0.),
  FOREIGN KEY(Computation_id) REFERENCES computation(Computation_id)
);

sqlite> INSERT INTO energy VALUES(1,1,-40.19873);
sqlite> INSERT INTO energy VALUES(2,2, 76.02686);
Error: constraint failed
```

DEFAULT

Default value to set when column value is omitted

```
CREATE TABLE log (
      id  INTEGER PRIMARY KEY AUTOINCREMENT,
    date  DATE DEFAULT (date()),
    time  DATE DEFAULT (time()),
message  TEXT NOT NULL
);
```

```
sqlite> INSERT INTO log(message) VALUES("first log message");
sqlite> INSERT INTO log(message) VALUES("second log message");
sqlite> INSERT INTO log(message) VALUES("third log message");
sqlite> select * from log;
id|date|time|message
1|2015-02-05|18:36:54|first log message
2|2015-02-05|18:36:59|second log message
3|2015-02-05|18:37:03|third log message
```

Triggers

```
CREATE TRIGGER log_computation INSERT ON computation
WHEN new.Method_id = 2
BEGIN
  INSERT INTO log(message) VALUES("Inserted a new MP2 calculation");
END;
```

```
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(1,1);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(1,2);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(2,2);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(3,2);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(3,1);
sqlite> INSERT INTO computation(Method_id,Mol_id) VALUES(2,2);

sqlite> select * from log;
1|2015-02-05|18:36:54|first log message
2|2015-02-05|18:36:59|second log message
```

```
3|2015-02-05|18:37:03|third log message
4|2015-02-05|18:48:21|Inserted a new MP2 calculation
5|2015-02-05|18:48:52|Inserted a new MP2 calculation
```

Views

View are virtual tables :

```
CREATE VIEW comp_view AS
  SELECT Method,Formula from computation
  NATURAL JOIN method
  NATURAL JOIN molecule;

sqlite3> select * from comp_view;
Hartree-Fock|CH4
Hartree-Fock|H2O
MP2|H2O
QMC|H2O
QMC|CH4
CCSD(T)|H2O
Hartree-Fock|CH4
Hartree-Fock|H2O
```

```
MP2|H2O
QMC|H2O
QMC|CH4
CCSD(T)|H2O
MP2|H2O
```