SQLITE

Amritha Nambiar & Dejun Qian

What is SQLite

- Open source embedded, Relational database implemented in pure, portable, ANSI C.
- Designed to plug directly into your programs, scripts, or web applications
- Supports a large subset of ANSI SQL.
- Supports transactions, views, indexes, triggers, subqueries, check constraints and a variety of other features found in relational databases.
- Variety of open source extensions allowing use with languages such as Perl, Python, Ruby, Java, PHP, .NET, Scheme, Smalltalk, Objective C, Delphi, Ada, Haskell etc
- It is designed to be small, portable, reliable, customizable, efficient, and free.

History

- The original implementation was designed by D. Richard Hipp
- Design goals of SQLite were to allow the program to be operated without a database installation or administration.
- In 2000 version 1.0 of SQLite was released. This initial release was based off of GDBM (GNU Database Manager)
- Version 2 followed within the year. By 2001, many open source projects were beginning to use it. Several languages had bindings for it.
- In 2004, SQLite had a major upgrade to version 3.
- Version 3.0 added many useful improvements such as manifest typing, more SQL features etc.

SQLite Users

- Apple: Safari, Mail
- Adobe: Uses SQLite in products like Photoshop and Acrobat\Adobe reader
- Google: Gears, SQLite is also used in the mobile OS platform- Android
- Mozilla: Firefox , SQLite is used in Firefox to store metadata.
- PHP - Php comes with SQLite 2 and 3 built in.
- Python- SQLite is bundled with the Python programming language.
- Nokia, Dlink, Philips, Palm, and *lots* of embedded products

SQLite Design Considerations

- Flexibility: It is easy to customize, modify, extend.
- Compactness: SQLite strives to stay small, and resists bloat at all costs. Big features are conditionally-compiled or dynamically loadable extensions. Core library will never outgrow the embedded space.
- Reliability
- Portability. Run everywhere. It is written in ANSI with an OS abstraction layer at the bottom
- Stability

SQLite Features

- No configuration. Just drop in the C library and go.
- No server process to administer or user accounts to manage.
- In memory databases. Databases don't have to exist on OS files. They can be created in memory.
- Virtual tables
- Easy to backup and transmit database (just copy the file)

SQLite Features

- Conflict resolution. SQLite has built in rules to overcome integrity violations. If you try to insert a record whose primary key value is already in the table, conflict resolution gives you the option of automatically overwriting the record, or failing. Conflict resolution can be defined at different levels ranging from the SQL command all the way down to the table definition.
- Attaching databases. You can attach multiple database files to a single connection and read to and from them as if they belonged to a single database.
- Manifest typing. SQLite is dynamically typed, and works a lot like a scripting language where other databases are statically typed. This can provide a great deal of flexibility in many of the same ways.

Limitations

- Query Optimizer. SQLite does not have a sophisticated query optimizer.
- Very large datasets - With the default page size of 1024 bytes, an SQLite database is limited in size to 2 terabytes (241 bytes). And even if it could handle larger databases, SQLite stores the entire database in a single disk file and many filesystems limit the maximum size of files to something less than this. .
- Access control – This can be a disadvantage for database systems that have large number of users and require security
- Client/Server Applications - SQLite will work over a network filesystem, but because of the latency associated with most network filesystems, performance will not be great. Also, the file locking logic of many network filesystems implementation contains bugs (on both Unix and Windows

SQLite Vs Commonly used RDBMS

SQLite	Commonly used RDBMS (Oracle, MYSQL etc)
• Easy to set up - no configuration or installation is necessary • Not suitable where user management is needed • Suitable for using in embedded applications • Not suitable where concurrency transactions on the databases is required • Not good for large scale databases as SQLite stores the database in a single file • Not readily scalable. Altering tables is not permitted in SQLite except for adding columns and renaming tables • Is not suited in a situation where Stored procedures are needed and where certain types of joins are needed	• Far more difficult to set up and configuration of users is a must • Suited for managing users and their permissions • Not suitable for embedding in some hardware as you would still need the server component of the database. • Perfect for concurrency transactions on the data and is well suited for multi-user environment • Great for large scale production applications which scale even over clustered database configurations • Highly scalable as far as the MySQL data and tables go and can be manipulated any time the MySQL DBA. • Fully compatible with stored procedures, triggers, view and other operations common with other major RDBMS

SQLite – Basic Commands

- A standalone program called sqlite3 is provided that can be used to create a database, define tables within it, insert and change rows, run queries and manage a SQLite database file. This program is a single executable file on the host machine. It also serves as an example for writing applications that use the SQLite library.

- .help show instructions
- .database show current database
- .table show all tables in current database
- .exit exit sqlite

- A valid sql statement must end with a semicolon “;” (above commands do not need)

SQL Commands

- Create Table
- Agents(agentid: integer, fname: string, lname: string)
- To create this table, using sql language as:

```
create table agent( agent_id INTEGER primary key,
fname varchar2(30),
lname varchar2(40)
);
```

- To insert a record into the table, using sql language as:

```
Insert into agents values (0,"John","Doe");
```

- To import data into table from csv file

```
.seperator "," ;
.import agents.csv agents
```

SQL Commands

PRIMARY KEY assigned as INTEGER will auto increment when null is inserted into the column

With this table, the statement

INSERT INTO agent VALUES(NULL, “first” , “last”);

is logically equivalent to saying:

INSERT INTO agent VALUES((SELECT max(id) FROM agent)+1, “First” , “Last”);

ALTER Table

SQLite has limited ALTER TABLE support that you can use to add a column to the end of a table or to change the name of a table. If you want to make more complex changes in the structure of a table, you will have to recreate the table.

e.g To remove column c from table t1

```
CREATE TEMPORARY TABLE t1_backup(a,b);
INSERT INTO t1_backup SELECT a,b FROM t1;
DROP TABLE t1;
CREATE TABLE t1(a,b);
INSERT INTO t1 SELECT a,b FROM t1_backup;
DROP TABLE t1_backup;
```

Demo on Android System

References

- http://www.sqlite.org/
- http://en.wikipedia.org/wiki/SQLite
- http://www.oracle.com/technetwork/products/berkeleydb/bdb-sqlite-comparison-wp-176431.pdf