
globsyn
Taking People To The Next Level

globsyn


globsyn
finishing school

Database - SQLite

Database – SQLite

- Topics to be covered in this session

  - Introduction to SQLite
  - SQLite open helper and creating datbase
  - Opening and closing database

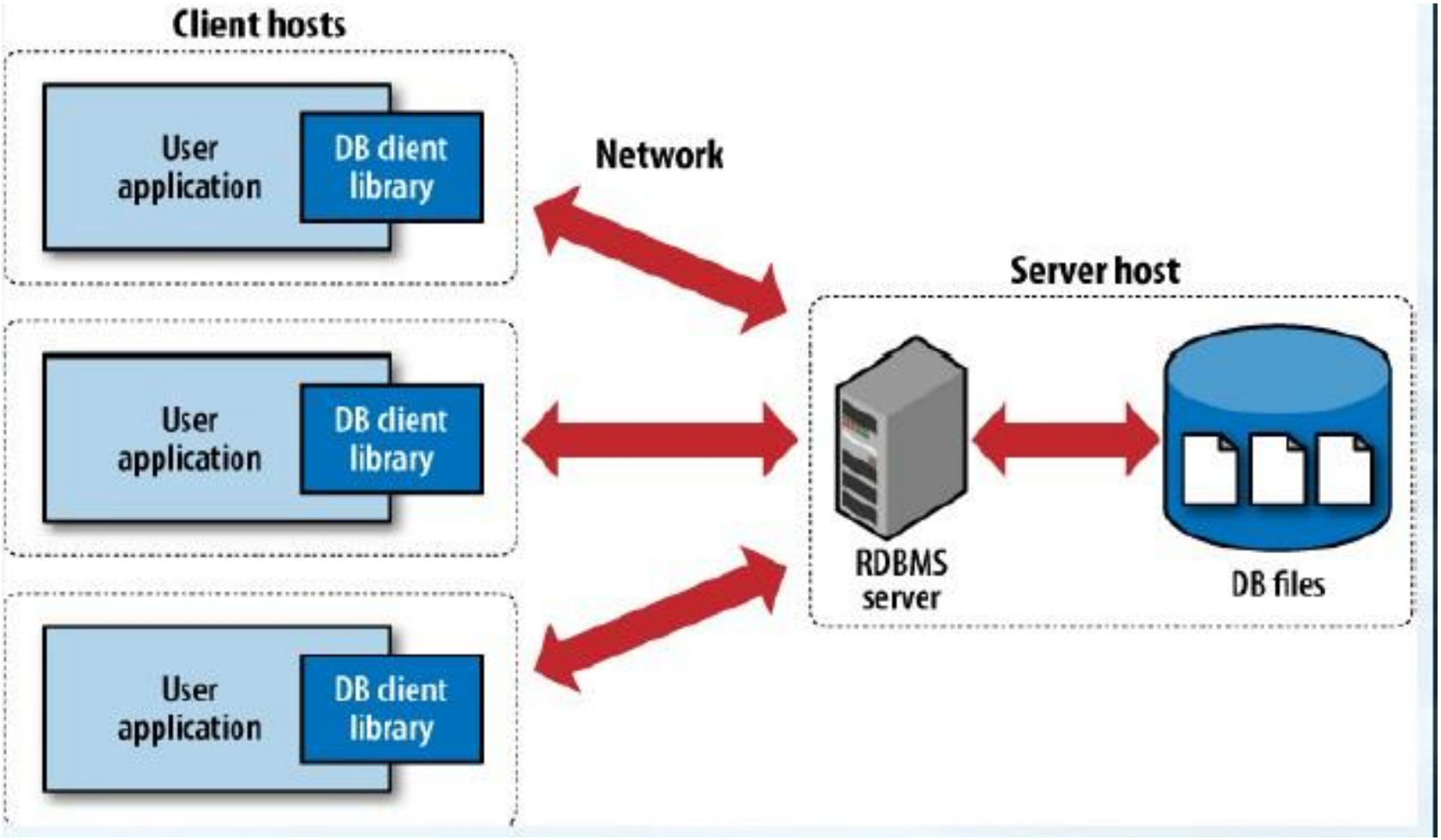
SQLite – What is it?

- It is a Relational Database Management System(RDBMS)
- It is a light weight SQL

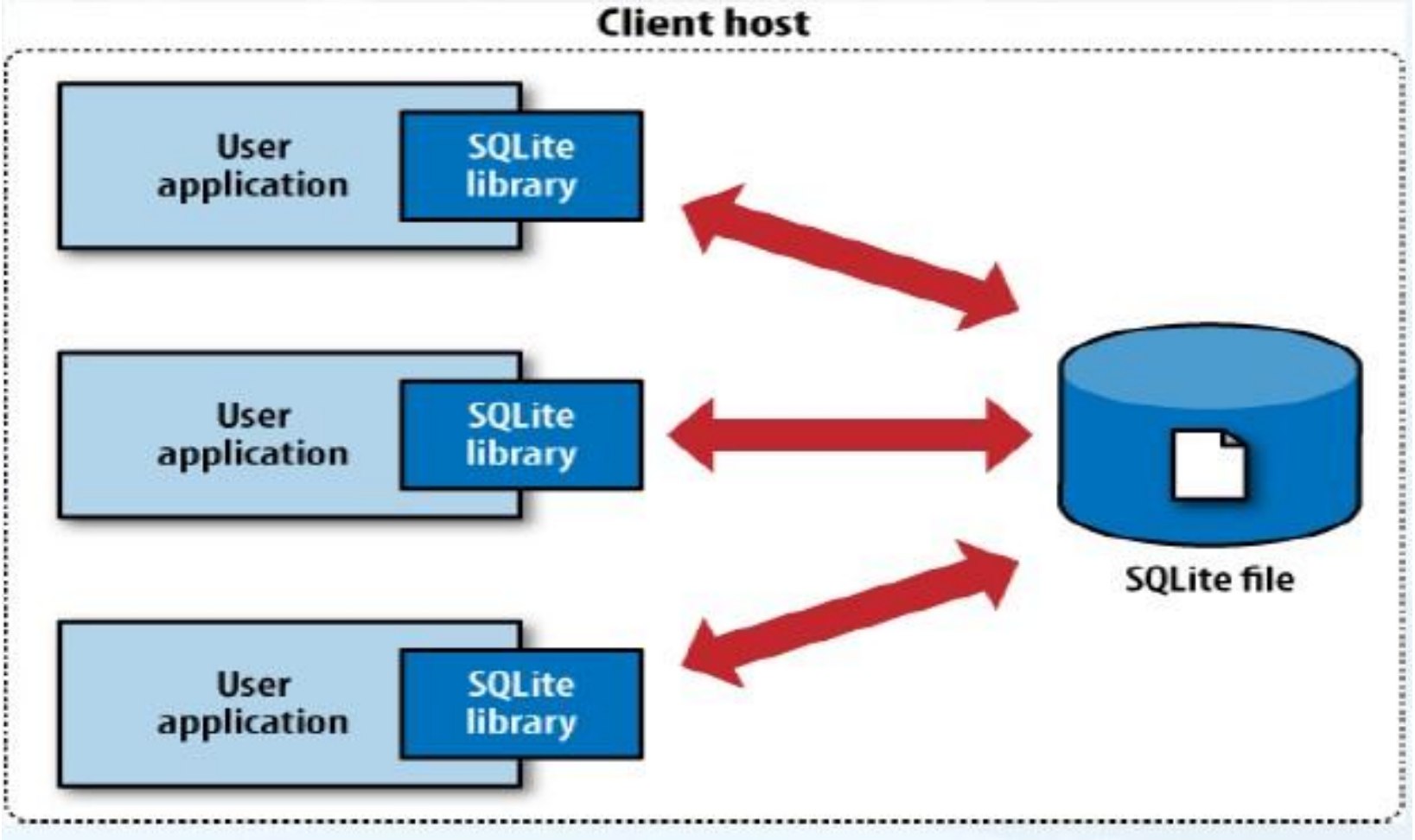
Features of SQLite

- Server less
- Zero configuration
- Cross platform
- Self contained
- Small run time foot print
- Transactional
- Full featured
- Highly reliable

Client Server Architecture



Server Less Architecture



Advantages of SQLite

- Application files
- Application cache
- Archives and data stores
- Client/Server stand-in
- Generic SQL engine
- Teaching tool

Disadvantages of SQLite

- High transaction rate
- Extremely large data sets
- Access control (Database Files)
- Client-server environment
- Replication

Installation of SQLite

- Step 1: Go to the source link http://sqlite.org/download.html
- Step 2: Download sqlite-autoconf-3071700.tar.gz
- Step 3: Extract the downloaded file
- Step 4: Open Command Prompt(in Windows) or Terminal(in Linux)
- Step 5: Go to the path where the downloaded file has been saved
- Step 6: Type make (enter)

SQLite : Ready to Run

- First Step to SQLite

```
[SKM@localhost ~]$ sqlite3 testdb1.db
SQLite version 3.7.17 2013-05-20    00:56:22
Enter ".help" for instructions
Enter SQL statements terminated with a
";"
Sqlite> _
```

Working with SQLite

- Single Line Command

```
sqlite> select 5,9;
5|9
```

- Multi Line Command

```
sqlite> select 5
...> ,9
...> ;
5|9
```

Working with SQLite Contd.

- Dot (.) Commands
- .help: Will list all dot (.) commands

Example: sqlite> select 5,9;

5|9

sqlite> .separator ,

sqlite> select 5,9;

5,9

Useful Dot Commands

- .header
- .mode
- .exit/.quit
- .width
- .output
- .import
- .schema
- .backup
- .read
- .show

Creating Tables In SQLite

Syntax

```
CREATE TABLE table_name
(
attrib_1 Column_Type Column_Constraints,
attrib_2 Column_Type Column_Constraints,
attrib_3 Column_Type Column_Constraints,
...
Table_Constraints
);
```

Creating Views in SQLite

Syntax

CREATE VIEW view_*name* AS Query Statement*;*

SQLite Column Types

- NULL
- Integer: Range {-9,223,372,036,854,775,808 to +9,223,372, 036,854,775,807}, or roughly 19 digits.
- Float: 8-byte, IEEE 754 double-precision number.
- Text : A variable-length string, stored using the database encoding (UTF-8, UTF-16BE, or UTF-16LE).
- BLOB: Binary Large Object. A length of raw bytes.

SQLite Column Constraints

```
CREATE TABLE parts
(
part_id INTEGER PRIMARY KEY,
stock INTEGER DEFAULT 0 NOT NULL,
desc TEXT CHECK( desc != '' )
);
```

SQLite Table Constraints

```
CREATE TABLE orders
(
part_id INTEGER NOT NULL,
vendor_id INTEGER NOT NULL,
qty INTEGER NOT NULL,
rate INTEGER NOT NULL,
PRIMARY KEY (part_id, vendor_id)
);
```

Drop vs Delete

DROP TABLE table_name;

DELETE * FROM table_name WHERE

condition(s);

Insert

- Syntax 1: INSERT INTO table_name (col1, col2, ..., coln)VALUES (val1, val2, ..., valn);
- Syntax 2: INSERT INTO table_nameVALUES (val1, val2, ..., valn);
- Syntax 3: INSERT INTO table_name (col1, col2, ..., coln) SELECT query_statement;

Update vs Alter

ALTER TABLE *table_name RENAME TO*

new_table_name;

ALTER TABLE *table_name ADD COLUMN*

column_def...;

UPDATE table_name SET column_name=new_value

[, ...] WHERE

expression;

General Syntax

- SELECT *(or Column Names separated by comma)
FROM table_name(or table names separated by comma)
WHERE condition(s)
ORDER BY column_name(or number)
GROUP BY
HAVING condition(on aggregate function);

- SELECT *(Column Names separated by comma)
FROM table_name(or table names separated by comma)
WHERE condition(s)
ORDER BY column_name(or number)
GROUP BY
HAVING condition(on aggregate function);

SQLite – Table Constarints

- +, -, *, /, % : Standard Arithmetic Operators
- ~ : Bitwise Inversion Operator
- || : Concatenation Operator
- <, <=, =>, > : Standard Comparison Operators
- =, == : Equality Operators
- !=, <> : Not Equality Operators
- AND, OR : Logic Operators

Sample DB Design

- Create the Database with following specifications
  Database Name: testdb1.db
  Tables: personalinfo(id#,name)
  enrollment(id#, sub)

- Test the Following Operations on testdb1.db
  INSERT, DELETE, UPDATE, DROP,
  ALTER, Run Queries

Create Tables

- CREATE TABLE personal info (id INTEGER PRIMARY KEY, name TEXT NOT NULL);
- CREATE TABLE enrollment(id INTEGER PRIMARY KEY, sub TEXT NOT NULL);

Insert Values

- INSERT INTO personalinfo

VALUES (1, 'Ram Mohon Sardar');

- INSERT INTO enrollment

VALUES (1, 'Physics');

Query

- SELECT * FROM personalinfo;
- SELECT * FROM enrollment;

Dot(.) Operators

- .headers on: Will Display the header row
- .mode column: Column Wise View
- .separator: Each value will be separated by a comma
- .width 3,20 : Column Width Configuration

DROP Table

- DROP TABLE personal info;
- DROP TABLE enrollment;

DELETE Table

- SELECT * FROM personalinfo;
- DELETE FROM personalinfo

  WHERE name = "Ram Mohon Sardar";
- SELECT * FROM personalinfo;

UPDATE Table

- SELECT * FROM personalinfo;
- UPDATE personalinfo

SET name = "Ram Krishna Das"

WHERE name = "Ram Mohon Sardar";

- SELECT * FROM personalinfo;

ALTER

- ALTER TABLE personalinfo RENAME TO pi;
- ALTER TABLE pi RENAME TO personalinfo;

.schema personalinfo

- ALTER TABLE personalinfo ADD address TEXT;

.schema personalinfo

Query

- SELECT *

FROM personalinfo

WHERE name='Sam Palit';

- SELECT *

FROM personalinfo

WHERE name="Sam Palit";

Query

- SELECT *

FROM enrollment

ORDER BY 2;

- SELECT *

FROM enrollment

ORDER BY sub;


Taking People To The Next Level ...