  
**What is SQLite?**

SQLite is an **Open Source database**. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. The database requires limited memory at runtime (approx. 250 KByte) which makes it a light weight database to embed into other runtimes.

SQLite supports the data types TEXT (like String in Java), INTEGER (like long in Java) and REAL (like double in Java).

All other types must be converted into one of these fields before getting saved in the database.

SQLite is different from most other SQL database engines in that its primary design goal is to be simple:

1. Simple to administer
2. Simple to operate
3. Simple to embed in a larger program
4. Simple to maintain and customize

**Features of SQLite**

1. **Zero configuration** – SQLite does not need to be installed as there is no setup procedure to use it.
2. **Serverless** – SQLite is not implemented as a separate server process.
3. **Stable Cross-Platform Database File** – The SQLite file format is cross-platform. A database file written on one machine can be copied to and used on a different machine with a different architecture.
4. **Single Database File** – A SQLite database is a single ordinary disk file that can be located anywhere in the directory hierarchy.
5. **Compact** – When optimized for size, the whole SQLite library with everything enabled is less than 400KB in size

**Advantages of using SQLite**

1. There is no file parsing and no need to generate code to read/write/update the file.
2. Content can be accessed and updated using powerful SQL queries, greatly reducing the complexity of the application code.
3. The content can be viewed using third-party tools like Toad.
4. The application file is portable across all operating systems, 32-bit and 64-bit and big- and little-endian architectures.
5. Content is updated continuously and atomically so that there is no work lost in the event of a power failure or crash.

### Well Known Users of SQLite

Adobe Mozilla Google McAfee Skype PHP Microsoft

**SQL Commands**

Create, Read, Update and Delete (CRUD) operations in SQLite DB

1. **Command to create a table**

**Syntax**

Create table <table-name> ( col-name data-type, col-name data-type, ….n);

SQLite supports the data types TEXT (like String in Java), INTEGER (like long in Java) and REAL (like double in Java).

**Example:**

Create table Empoyee (Id INTEGER, Name TEXT, Salary REAL);

1. **Command to insert data into a table**

**Syntax**

Insert into <table-name> values (val1,val2,val3,…..n);

**Example**

Insert into Employee values (1212, ’Adwaith’, 1500000.00);

**Note: - TEXT** must be enclosed in single quotations.

1. **Command to Update data in a table**

**Syntax**

Update <table-name> set col1 = value, col2 = value, ….n where condition;

**Example**

Update Employee set Salary=1800000.00 where id=1212;

1. **Command to Delete data from a table**

**Syntax**

Delete from <table-name> where condition;

**Example**

Delete from Employee where id=1212;

1. **Command to Read set of data from a table**

**Syntax**

Select \* from <table-name> ;

**Example**

Select \* from Employee;

**Syntax**

**Select \* from <table-name> where condition;**

**Example**

Select \* from Employee where id = 1212;

**Syntax**

Select <col1>, <col2> from <table-name> where condition;

**Example**

Select Name, Salary from Employee where Id = 1212;

**SQLiteOpenHelper class (public abstract class)**

The **android.database.sqlite.SQLiteOpenHelper** class is used for database creation and version management.

For performing any database operation, you have to provide the implementation of **onCreate()** and **onUpgrade()** methods of SQLiteOpenHelper class.

**Constructor of SQLiteOpenHelper class**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)** | Creates an object for creating, opening and managing the database. |
|  |  |

**Methods of SQLiteOpenHelper class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| **public abstract void onCreate(SQLiteDatabase db)** | Called only once when database is created for the first time. |
| **public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)** | Called when database needs to be upgraded. |
| **public synchronized void close ()** | Closes the database object. |
| **public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)** | Called when database needs to be downgraded. |

**SQLiteDatabase class**

It contains methods to be performed on sqlite database such as create, update, delete, select etc.

**Method of SQLiteDatabase class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| **void execSQL(String sql)** | Executes the sql query not select query. |

**Java Snippet to creating a table (src/package/filename.java)**

**public** **class** DBClass **extends** SQLiteOpenHelper

{

**static** String *DATABASE\_NAME* = "Sample";

**static** **int** *DATABASE\_VERSION* = 1;

**public** DBClass(Context context)

{

**super**(context, *DATABASE\_NAME*, **null**, *DATABASE\_VERSION*);

}

@Override

**public** **void** onCreate(SQLiteDatabase db)

{

String qry = "create table tableone(id integer,name text,avg real,total integer)";

db.execSQL(qry);

}

@Override

**public** **void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion)

{

// **TODO** Auto-generated method stub

}

}

**Java Snippet (src/package/MainActivity.java)**

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

DBClass d= **new** DBClass(getApplicationContext());

}

**Java Snippet to inserting a row into table (src/package/filename.java)**

**public** **void** insertData()

{

SQLiteDatabase db = **this**.getWritableDatabase();

String qry = "insert into tableone values(1212,'Ravi',90.25,456)";

db.execSQL(qry);

qry = "insert into tableone values(1213,'Kumar',80.25,400)";

db.execSQL(qry);

qry = "insert into tableone values(1214,'Krishna',70.25,886)";

db.execSQL(qry);

qry = "insert into tableone values(1215,'Mohan',60.25,356)";

db.execSQL(qry);

}

**Note: -** User defined method need to be called explicitly by using reference.

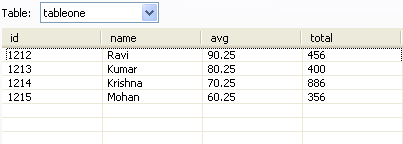
**Example: -** d.insertData();

**getWritableDatabase ()**

Create and/or open a database that will be used for reading and writing. The first time this is called, the database will be opened and **onCreate(SQLiteDatabase)**, **onUpgrade(SQLiteDatabase, int, int)** will be called.

Once opened successfully, the database is cached, so you can call this method every time you need to write to the database. (Make sure to call [close()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#close()) when you no longer need the database.)

After inserting rows to table

****

**Java Snippet to update a row in a table (src/package/filename.java)**

**public** **void** updateData()

{

SQLiteDatabase db = **this**.getWritableDatabase();

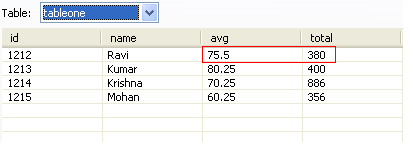
String qry = "update tableone set avg = 75.5, total=380 where id=1212";

db.execSQL(qry);

}

**Note: -** User defined method need to be called explicitly by using reference.

**Example: -** d.updateData();

****

**Java Snippet to delete a row in table (src/package/filename.java)**

**public** **void** deleteData()

{

SQLiteDatabase db = **this**.getWritableDatabase();

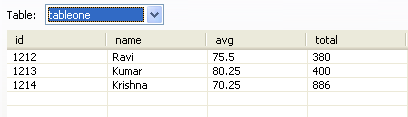
String qry = "Delete from tableone where id = 1215";

db.execSQL(qry);

}

**Note: -** User defined method need to be called explicitly by using reference.

**Example: -** d.deleteData();

****

**Java Snippet to View rows from table (src/package/filename.java)**

**public** String getAllContacts()

{

String values = "Detail = ";

// Select All Query

String selectQuery = "SELECT \* FROM tableone";

SQLiteDatabase db = **this**.getWritableDatabase();

Cursor cursor = db.rawQuery(selectQuery, **null**);

// looping through all rows and adding to list

**if** (cursor.moveToFirst()) {

**do** {

**int** id = cursor.getInt(0);

String name = cursor.getString(1);

**float** avg = cursor.getFloat(2);

**int** total = cursor.getInt(3);

values= values+"-"+id+"-"+name+"-"+avg+"-"+total+"----";

} **while** (cursor.moveToNext());

}

**return** values; }

**Cursor**

**android.database.Cursor** This interface provides random read-write access to the result set returned by a database query.

**rawQuery (String sql, String[] selectionArgs)**

Runs the provided SQL and returns a **Cursor** over the result set.