Unit 3

SQLite Database

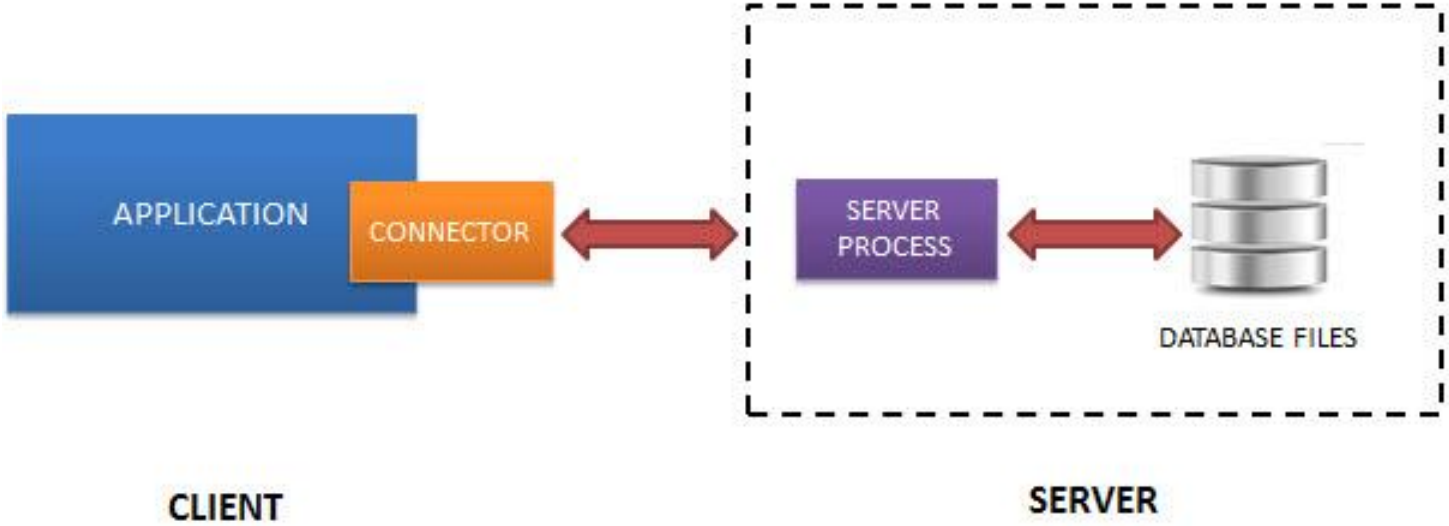
SQLite

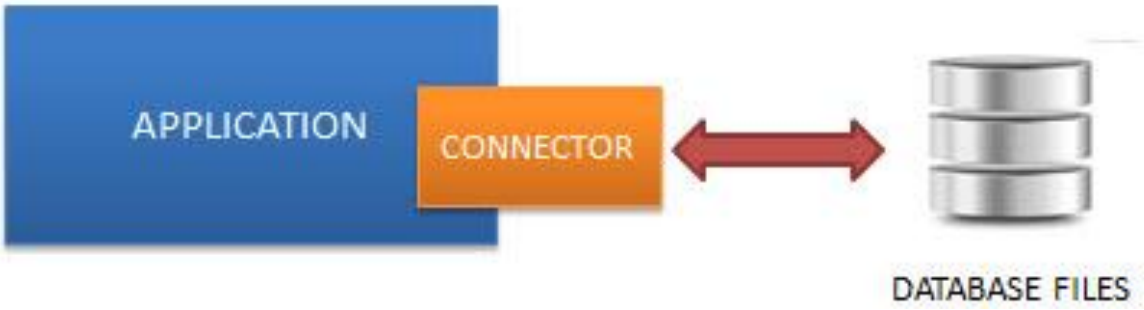
- SQLite is a open source SQL database that stores data to a text file on a device.
- Android comes in with built in SQLite database implementation.
- SQLite supports most relational database features.
- In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC

SQLite

- SQLite database is integrated with all android applications & accessible by all classes within it.
- SQL is a programming language designed especially for managing data in relational databases.
- Unlike, RDBMS such as MySQL, PostgreSQL which require a separate server process to operate, SQLite does NOT require a server to run.
- The applications interact with the SQLite database read and write directly from the database files stored on disk.

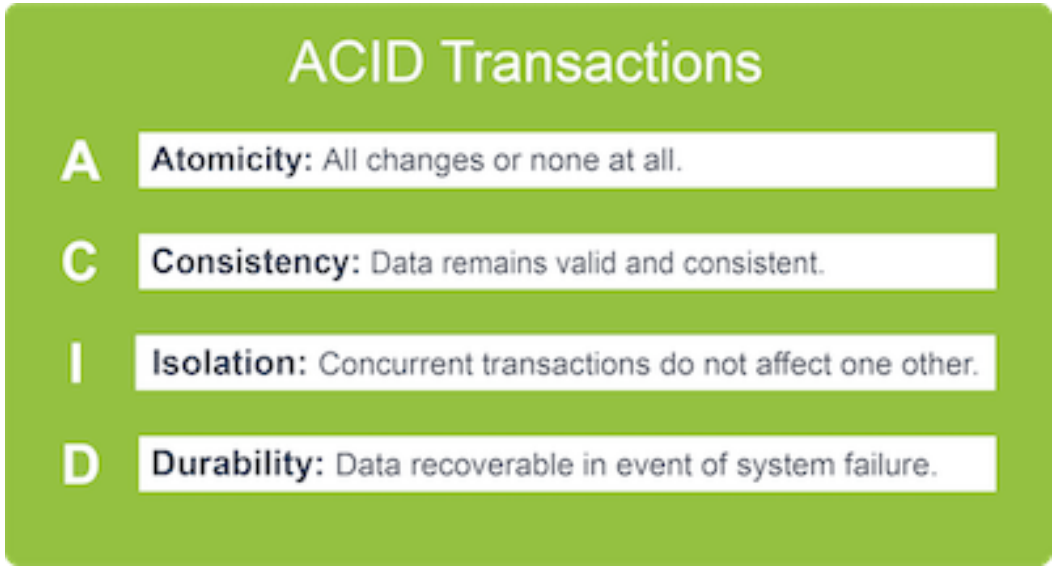
Server oriented vs Server-less





Features of SQLite

- SQLite requires minimal support
- don't need to "install" SQLite before using it
- All transactions in SQLite are fully ACID-compliant



Other Features of SQLite

- SQLite uses dynamic types for tables i.e. you can store any value in any column, regardless of the data type.
- SQLite allows a single database connection to access multiple database files simultaneously.
- This brings many nice features like joining tables in different databases or copying data between databases in a single command.
- SQLite is capable of creating in-memory databases which are very fast to work with.

Unsupported Features

- Stored Procedures
- Right Outer Join and Full Outer Join
- The RENAME TABLE and ADD COLUMN variants of the ALTER TABLE command are supported. The DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT are not supported.

Unsupported Features

- FOR EACH ROW triggers are supported but not FOR EACH STATEMENT triggers.
- VIEWs in SQLite are read-only. You may not execute a DELETE, INSERT, or UPDATE statement on a view.
- The only access permissions that can be applied are the normal file access permissions of the underlying operating system

Database Implementation

Database Packages

android.database.sqlite

It contains the classes to manage your own databases

android.database.sqlite.SQLiteDatabase

Exposes methods to manage a SQLite database.

SQLiteDatabase has methods to create, delete, execute SQL commands, and perform other common database management tasks.

Steps

1. Create "SQLiteDatabase" object.
2. Open or Create database and create connection.
3. Perform insert, update or delete operation.
4. Create Cursor to display data from table of database.
5. Close the database connectivity.

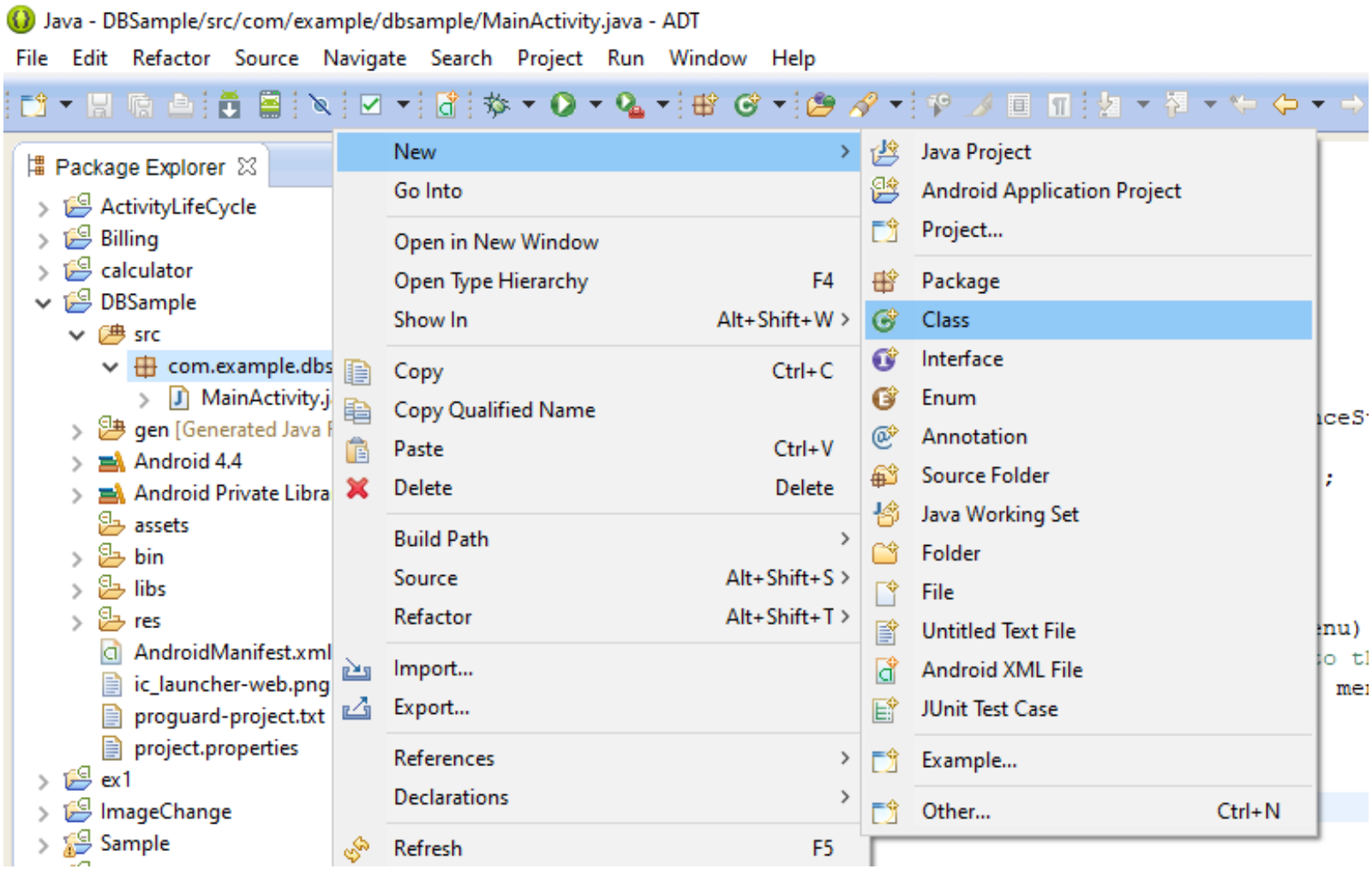
Database Creation

- **openOrCreateDatabase()**
- Takes database name and mode as parameters
- It returns an instance of SQLite database
- SQLiteDatabase mydatabase = openOrCreateDatabase("your database name", MODE_PRIVATE, null);

Method & Description
openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler) This method only opens the existing database with the appropriate flag mode. The common flags mode could be OPEN_READWRITE OPEN_READONLY
openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags) It is similar to the above method as it also opens the existing database but it does not define any handler to handle the errors of databases
openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory) It not only opens but create the database if it not exists. This method is equivalent to openDatabase method.
openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory) This method is similar to above method but it takes the File object as a path rather then a string. It is equivalent to file.getPath()

SQLiteOpenHelper

- **android.database.sqlite.SQLiteOpenHelper**
- A helper class to manage database creation and version management.
- This class takes care of opening the database if it exists, creating it if it does not, and upgrading it as necessary.
- You create a subclass implementing onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) and optionally onOpen(SQLiteDatabase)
- Transactions are used to make sure the database is always in a sensible state.


Java - DBSample/src/com/example/dbsample/MainActivity.java - ADT
File Edit Refactor Source Navigate Search Project Run Window Help
Package Explorer
ActivityLifeCycle
Billing
calculator
DBSample
src
com.example.dbs
MainActivity.j
gen [Generated Java F
Android 4.4
Android Private Libra
assets
bin
libs
res
AndroidManifest.xml
ic_launcher-web.png
proguard-project.txt
project.properties
ex1
ImageChange
Sample
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
References
Declarations
Refresh F5
Java Project
Android Application Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
Android XML File
JUnit Test Case
Example...
Other... Ctrl+N

activity_main.xml | MainActivity.java | *Dbsample.java

```
package com.example.dbdemo;


public class Dbsample extends SQLiteOpenHelper{

}
```

activity_main.xml | MainActivity.java | *Dbsample.java

```
package com.example.dbdemo;


public class Dbsample extends SQLiteOpenHelper{
```

ackage com.example.dbdemo;

nport android.database.sqlite.SQLiteOpenHelper;

Press 'Tab' from proposal table or click for focus

- Import 'SQLiteOpenHelper' (android.database.sqlite)
- Create class 'SQLiteOpenHelper'
- Rename in file (Ctrl+2, R)
- Fix project setup...

activity_main.xml | MainActivity.java | *SQLiteHelper.java

```
package com.example.dbsample;

import android.database.sqlite.SQLiteOpenHelper;

public class SQLiteHelper extends SQLiteOpenHelper{

}
```

Implicit super constructor SQLiteOpenHelper() is undefined for default constructor. Must define an explicit constructor

2 quick fixes available:

- Add constructor 'SQLiteHelper(Context,String,CursorFactory,int)'
- Add constructor 'SQLiteHelper(Context,String,CursorFactory,int,DatabaseErrorHandler)'

activity_main.xml | MainActivity.java | *SQLiteHelper.java

```
package com.example.dbsample;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class SQLiteHelper extends SQLiteOpenHelper{

    public SQLiteHelper(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);
        // TODO Auto-generated constructor stub
    }

}
```

activity_main.xml | MainActivity.java | *Dbsample.java

```
package com.example.dbdemo;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class Dbsample extends SQLiteOpenHelper{

    public Db
            i
        super
        // TO
    }

}
```

- Add unimplemented methods
- Make type 'Dbsample' abstract
- Rename in file (Ctrl+2, R)
- Rename in workspace (Alt+Shift+R)

2 methods to implement:
- android.database.sqlite.SQLiteOpenHelper.o
- android.database.sqlite.SQLiteOpenHelper.o

Press 'Tab' from proposal tab

activity_main.xml | MainActivity.java | Dbsample.java

```java
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class Dbsample extends SQLiteOpenHelper{



    public Dbsample(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);
        // TODO Auto-generated constructor stub
    }

    @Override
    public void onCreate(SQLiteDatabase arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
        // TODO Auto-generated method stub
```

activity_main.xml | MainActivity.java | *Dbsample.java

```java
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class Dbsample extends SQLiteOpenHelper{

    public static final String DATABASE_NAME="testdb";
    public static final int DATABASE_VERSION=1;
    public static final String TABLE_NAME="student";
    public static final String S_ID="s_id";
    public static final String S_NAME="s_name";

    public Dbsample(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);
        // TODO Auto-generated constructor stub
    }

    @Override
    public void onCreate(SQLiteDatabase arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
        // TODO Auto-generated method stub
```

activity_main.xml | MainActivity.java | *Dbsample.java

```
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class Dbsample extends SQLiteOpenHelper{

    public static final String DATABASE_NAME="testdb";
    public static final int DATABASE_VERSION=1;
    public static final String TABLE_NAME="student";
    public static final String S_ID="s_id";
    public static final String S_NAME="s_name";

    public Dbsample(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);
        // TODO Auto-generated constructor stub
    }

    @Override
    public void onCreate(SQLiteDatabase arg0) {
        // TODO Auto-generated method stub
        arg0.exe
    }

    @Override
    public void                                    arg1, int arg2) {
        // TODO
```

execSQL(String sql) : void - SQLiteDatabase
execSQL(String sql, Object[] bindArgs) : void - SQLiteDatabas

activity_main.xml | MainActivity.java | Dbsample.java

```
t android.database.sqlite.SQLiteDatabase.CursorFactory;
t android.database.sqlite.SQLiteOpenHelper;

c class Dbsample extends SQLiteOpenHelper{

ublic static final String DATABASE_NAME="testdb";
ublic static final int DATABASE_VERSION=1;
ublic static final String TABLE_NAME="student";
ublic static final String S_ID="s_id";
ublic static final String S_NAME="s_name";

ublic Dbsample(Context context, String name, CursorFactory factory,..

Override
ublic void onCreate(SQLiteDatabase arg0) {
    // TODO Auto-generated method stub
    arg0.execSQL("CREATE TABLE "+TABLE_NAME+"("+S_ID+" INTEGER PRIMARY KEY AUTOINCREMENT, "+S_NAME+" VARCHAR(50)");


Override
ublic void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {
    // TODO Auto-generated method stub
```

activity_main.xml | *MainActivity.java | Dbsample.java

```java
import android.database.sqlite.SQLiteOpenHelper;

public class Dbsample extends SQLiteOpenHelper{

    public static final String DATABASE_NAME="testdb";
    public static final int DATABASE_VERSION=1;
    public static final String TABLE_NAME="student";
    public static final String S_ID="s_id";
    public static final String S_NAME="s_name";

    public Dbsample(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);
        // TODO Auto-generated constructor stub
    }

    //creating myown constructor
    public Dbsample(Context context){
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase arg0) {
        // TODO Auto-generated method stub
        arg0.execSQL("CREATE TABLE "+TABLE_NAME+"("+S_ID+" INTEGER PRIMARY KEY AUTOINC
    }
```

activity_main.xml | MainActivity.java | Dbsample.java

```java
package com.example.dbdemo;

import android.os.Bundle;

public class MainActivity extends Activity {

    Dbsample samp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

activity_main.xml | MainActivity.java | Dbsample.java

```java
package com.example.dbdemo;

import android.os.Bundle;

public class MainActivity extends Activity {

    Dbsample samp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        samp = new Dbsample(this);
    }

    @Override
```

activity_main.xml | *MainActivity.java | Dbsample.java

```java
package com.example.dbdemo;

import android.os.Bundle;

public class MainActivity extends Activity {

    Dbsample samp;
    SQLiteDatabase db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        samp = new Dbsample(this);
        db=samp.getWritableDatabase();
    }
```

Database Insertion, Reading, Updating & Deletion

2 methods for all the operations

1. **parameterized queries** (Recommended): These are those queries which are performed using inbuilt functions (insert, query, update, delete) provided in SQLiteDatabase class.
2. **raw queries**: write simple sql queries as string and passed to rawQuery or execSQL.

Insertion through ContentValues

activity_main.xml | MainActivity.java | Dbsample.java

```
import android.os.Bundle;

public class MainActivity extends Activity {

    Dbsample samp;
    SQLiteDatabase db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        samp = new Dbsample(this);
        db=samp.getWritableDatabase();

        ContentValues cv = new ContentValues();
        cv.put(Dbsample.S_NAME, "Geetha");

        db.insert(Dbsample.TABLE_NAME, Dbsample.S_NAME, cv);
    }
```

Insertion through execSQL() method

activity_main.xml | *MainActivity.java | SQLiteHelper.java

```
SQLiteDatabase db;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    dbhelper=new SQLiteHelper(this);
    db=dbhelper.getWritableDatabase();

    //ContentValues cv = new ContentValues();
    //cv.put(SQLiteHelper.STUDNAME,"Eliza");

    //db.insert(SQLiteHelper.TABLE_NAME, SQLiteHelper.STUDNAME, cv);

    String insertquery = "INSERT INTO "+SQLiteHelper.TABLE_NAME+" ("+SQLiteHelper.STUDNAME+") VALUES ('Eliza')";
    db.execSQL(insertquery);

    //INSERT INTO student (studname) VALUES ('Eliza');
}
```

Fetching

- We can retrieve anything from database using an object of the Cursor class.

activity_main.xml | MainActivity.java | Dbsample.java

```
//method1 to fetch data
Cursor c1 = db.query(Dbsample.TABLE_NAME, new String[] {Dbsample.S_ID, Dbsample.S_NAME},
        null, null, null, null, null);
while(c1.moveToNext()){
    int id = c1.getInt(0);
    String name = c1.getString(1);
    Log.i("LOG_TAG", "ROW " + id + " HAS NAME " + name);
}
c1.close();

//method2 to fetch data
String selectQuery = "SELECT * FROM " +Dbsample.TABLE_NAME;
Cursor c2 = db.rawQuery(selectQuery, null);
while(c2.moveToNext()){
    int id = c2.getInt(c2.getColumnIndex(Dbsample.S_ID));
    String name = c2.getString(c2.getColumnIndex(Dbsample.S_NAME));
    Log.i("LOG_TAG", "ROW " + id + " HAS NAME " + name);

}
c2.close();
```

Updating & Deleting

```
//Item is a class representing any item with id, name and description
public void updateItem(Item item) {
  SQLiteDatabase db = getWritableDatabase();
  ContentValues contentValues = new ContentValues();
  contentValues.put("id", item.id);
  contentValues.put("name", item.name);
  contentValues.put("description", item.description);
  String whereClause = "id=?";
  String whereArgs[] = {item.id.toString()};
  db.update("Items", contentValues, whereClause, whereArgs);
}
```

```
public void deleteItem(Item item) {
SQLiteDatabase db = getWritableDatabase();
String whereClause = "id=?";
String whereArgs[] = {item.id.toString()};
db.delete("Items", whereClause, whereArgs);
}
```