SQLITE

WHY USE SQLITE?

As the name implies, SQLite is *light*weight

Android provides a robust class (SQLiteCursor) for interfacing with SQLite Databases

Relational databases are essential for effeciently querying complex data models

Android provides an Adapter (CursorAdapter) to expose data from a Cursor to a widget

OVERVIEW

1. Subclass SQLiteOpenHelper and define Schema and Migrations
2. Create a Data Source Helper class to encapsulate opening, closing, adding, deleting, and querying the DataBase
3. Subclass CursorAdapter to expose database data to Widgets

SQLITEOPENHELPER

- Helper class to manage database creation and version management
  - @Override Public Constructor, onCreate, and onUpgrade methods

- Constructor: pass context as a parm
- onCreate: establish database schema
- onUpgrade: execute migration code when changing database versions

```
public class PersonOpenHelper extends SQLiteOpenHelper {
    public PersonOpenHelper(Context context){
        super(context,DB_NAME, null, DB_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DB_CREATE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
     //Handle DB Migrations
    }
}
```

DATASOURCE HELPER CLASS

1. Open Database: call getWritableDatabase on instance of SQLiteOpenHelper (return SQLiteDatabase)
2. Close Database: call close on SQLiteDatabase instance
3. Encapsulate CRUD logic (Create, Read, Update, Delete)

CREATE

```
public void insertPerson(Person person) {

    mDatabase.beginTransaction();

    try {

            //Store a set of values for Table
            ContentValues values = new ContentValues();
            values.put(PersonOpenHelper.COLUMN_NAME, person.getName());
            values.put(PersonOpenHelper.COLUMN_PHONE, person.getPhone());

            //Insert item in database

            mDatabase.insert(PeopleOpenHelper.TABLE_PEOPLE, null, values);

            mDatabase.setTransactionSuccessful();
        } finally {
        mDatabase.endTransaction();
    }
}
```

READ

```
public Cursor selectPerson(String name){
        String whereClause = PersonOpenHelper.COLUMN_NAME + " == ?"; // ? = use select args

        Cursor cursor = mDatabase.query(
                PersonOpenHelper.TABLE_PERSON, //table
                new String[]{
                        PersonOpenHelper.COLUMN_NAME,
                        PersonOpenHelper.COLUMN_PHONE
                },
                whereClause,//selection
                new String[]{String.valueOf(name)},//selection args
                null,
                null,
                null
        );
        return cursor;
    }
```

UPDATE

```
public int updatePerson(Person person){
        String whereClause = PersonOpenHelper.COLUMN_NAME + " == ?";

        int rowsupdated = 0;

        ContentValues values = new ContentValues();
                values.put(PersonOpenHelper.COLUMN_NAME, person.getName());
                values.put(PersonOpenHelper.COLUMN_PHONE, person.getPhone());

        mDatabase.update(
                personOpenHelper.TABLE_PERSON, //table name to update in
                values, // map of column names
                whereClause, //which columns will be updated
                new String []{String.valueOf(hour.time)} // where args
        );

        return rowsupdated;
    }
```

DELETE

```
public int delete(int personID){
        String whereClause = PersonOpenHelper.COLUMN_ID + " == ?";
        return mDatabase.delete(PersonOpenHelper.TABLE_Person,
                                whereClause,
                                new String[]{String.valueOf(personID)}
                                );
    }
```

CURSOR

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

UPDATE LIST WITH CURSOR

```
protected void updateList(Cursor cursor){
    mPeople.clear();
    cursor.moveToFirst();
    while (!cursor.isAfterLast()){
        int i = cursor.getColumnIndex(PeopleOpenHelper.COLUMN_NAME);
        String name = cursor.getString(i);
        mPeople.add(new person(name));
        cursor.moveToNext();
    }
    ArrayAdapter<Person>adapter = new ArrayAdapter<Person>(this, R.layout.simple_list_item, mPeople);

    setListAdapter(adapter);
}
```