Android SQLite

What is SQLite?

SQLite is an open source database that supports relational database features like SQL syntax, transactions and prepared statements.

SQLite is requires limited memory at runtime approximately 250kb

SQLite in Android

SQLite is comes with every Android device.

Using an SQLite database in Android does not require a setup procedure or administration of the database.

Access to an SQLite database involves accessing the file system, this can be slow and therefore it is recommended to perform database operations asynchronously.

Creating and Updating DB

To create and upgrade a database in your Android application you create a subclass of the SQLiteOpenHelper i.e. a class extending SQLiteOpenHelper class.

```
public class DatabaseHelper extends SQLiteOpenHelper {
```

Create a constructor in the subclass and then invoke the super method then define database name and current version.

```
public DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
```

Creating and Updating DB Cont'd

We override the methods below in the subclass:

onCreate() :- called by the framework, if the database is accessed but not yet created.

onUpgrade() :- called if the database version is increased e.g. from 1 to 2

The SQLiteOpenHelper class provides getReadableDatabase() and getWriteableDatabase() methods to get an SQLiteDatabase object to either read or write

SQLiteDatabase

SQLiteDatabase is the base class for working with SQLite database in Android and provides methods to open, query, update and close database.

Methods provided by SQLiteDatabase: insert(), update() and delete();

To execute SQL statements we use execSQL() method.

To insert data, we package the data in a key value format using ContentValues object where the key represents the table column identifier and the value represents the content for the table record in the column.

SQLiteDatabase Cont'd

Example of packaging data using ContentValues object.

```
public boolean createCounty(String county_name, String county_governour, String desc) {
    ContentValues values = new ContentValues();
    values.put("county_name", county_name);
    values.put("county_governour", county_governour);
    values.put("county_desc", desc);

    long insertedId = 0;
    insertedId = database.insert(dbHelper.TABLE_COUNTIES, null,
            values);
    if (insertedId != 0) {
        return true;
    }

    return false;
}
```

SQLiteDatabase Cont’d

Queries can be created using rawQuery and query methods or SQLiteQueryBuilder class.

rawQuery() :- directly accepts an SQL select statement as input.

query() :- provides a structured interface for specifying the SQL query.

SQLiteQueryBuilder :- helps to build SQL queries.

rawQuery() Example

Cursor cursor = getReadableDatabase().rawQuery("select * from cars where _id = ?", new String[] {id});

query() Example

```
public ArrayList<CountyModel> fetchAllCounties() {
    ArrayList<CountyModel> counties = new ArrayList<CountyModel>();

    Cursor cursor = database.query(DatabaseHelper.TABLE_COUNTIES,
            null, null, null, null, null, null);

    cursor.moveToFirst();
    while (!cursor.isAfterLast()) {
        CountyModel model = new CountyModel(cursor.getString(1), cursor.getString(2), cursor.getString(3));
        counties.add(model);
        cursor.moveToNext();
    }
    // make sure to close the cursor
    cursor.close();
    return counties;
}
```

Parameters of query method

String dbName	The table name that you want to query.
String[] columnNames	A list of which table columns to return. Passing “null” will return all columns
String whereClause	Selection of data, null will select all data
String[] selectionArgs	You may include “?” in the “whereClause”, the question mark (s) will get replaced by the values from the selectionArgs array
String[] groupBy	A filter declaring how to group rows, null will cause the rows not to be grouped
String[] having	Filter for the groups, null means no filter
String[] orderBy	Order data e.g. by DESC, null means no order

Cursor

A query returns a Cursor object.

```
Cursor cursor = database.query(DatabaseHelper.TABLE_COUNTIES,
        null, null, null, null, null, null);
```

To get the number of elements of the resulting query use the getCount() method.

Cursor Cont'd

To traverse through the cursor, we use moveToFirst() and moveToNext() methods.

The isAfterLast() method allows to check if the end of the query has been reached.

Cursor provides get*() methods e.g. getString(columnIndex) to access the column data at the current position of the result

```
while (!cursor.isAfterLast()) {
    CountyModel model = new CountyModel(cursor.getString(1), cursor.getString(2), cursor.getString(3));
    counties.add(model);
    cursor.moveToNext();
}
```

NB: A cursor needs to be closed with the close() method call.

Resourceful Links

Try out the tutorial: Using SQLite
http://www.vogella.com/tutorials/AndroidSQLite/article.html

Read more about SQLite - Android
http://www.tutorialspoint.com/android/android_sqlite_database.htm