Program Database Sederhana di Android

Desember 2011
Oleh : Feri Djuandi

Tingkat: [√] Pemula [ ] Menengah [ ] Mahir

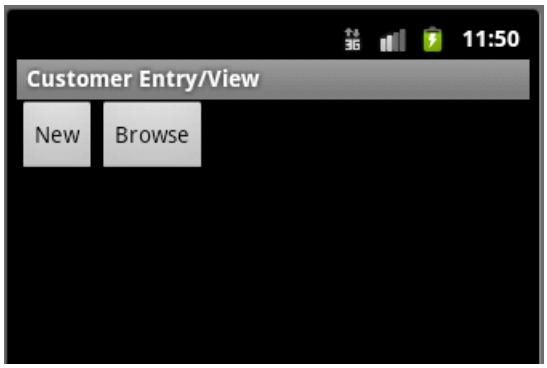
Platform : Android 2.3, Eclipse, SQLite

Artikel ini merupakan lanjutan dari tulisan sebelumnya yang berjudul “Menggunakan Database di Android”. Anda dianjurkan untuk membaca artikel tersebut karena mengandung penjelasan dasar-dasar database di Android yang sangat berguna untuk memberikan landasan pemahaman terutama untuk pemula.

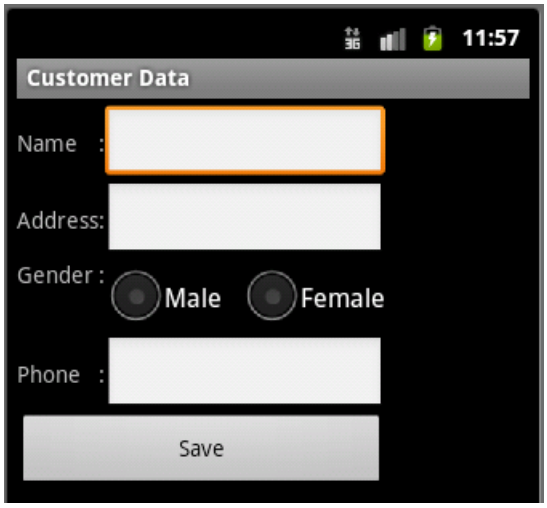
Pada pembahasan ini akan diperlihatkan sebuah program sederhana untuk membuat database **SQLite**, membuat sebuah table dan mengisi data ke dalamnya, dan menampilkan data tersebut. Penjelasan ini sengaja menggunakan contoh yang sangat sederhana supaya para pembaca dapat memahami prinsip-prinsip yang esensial di dalam penggunaan database SQLite. Tampilan output pada layar pun akan dibuat minimalis dengan mengabaikan tampilan yang dipercantik untuk menghindari tambahan-tambahan baris program pemanis yang mengesankan keruwetan dan mengaburkan fokus pembaca pada inti pembahasan.

Menyimpan dan menampilkan data adalah operasi database yang sangat mendasar, namun perlu disadari bahwa aplikasi yang canggih dan sangat kompleks sekalipun tidak lepas dari operasi-operasi dasar tersebut. Oleh karena itu para programmer perlu mengetahui teknik-teknik dan penggunaan *class-class* yang berkaitan dengan database mulai dari tahap yang awal. Setelah dikuasainya konsep dasar tersebut, selanjutnya Anda bisa mengembangkan dan mengombinasikan berbagai teknik yang lebih baik dan efisien sesuai dengan keperluan.

Contoh program ini terdiri dari dua bagian, yaitu program yang menampilkan sebuah form pengisian data dimana pengguna bisa memasukkan data kemudian menyimpannya di dalam database. Bagian yang ke-dua adalah untuk menampilkan data yang tersimpan. Sebelum kedua bagian program itu bisa bekerja, tentunya program utama akan terlebih dahulu menyiapkan database dan membuat table dengan struktur tertentu. Untuk memberikan gambaran yang lebih jelas dari cara kerja program tersebut, silakan memperhatikan beberapa ilustrasi di bawah ini.

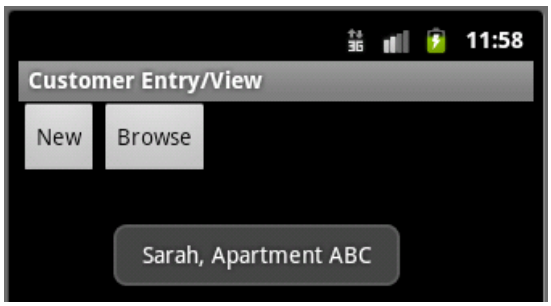


Saat program dijalankan ada dua buah tombol untuk memilih antara memasukan data (New) atau menampilkan data (Browse).



Saat tombol New ditekan, sebuah form ditampilkan untuk pengisian data.

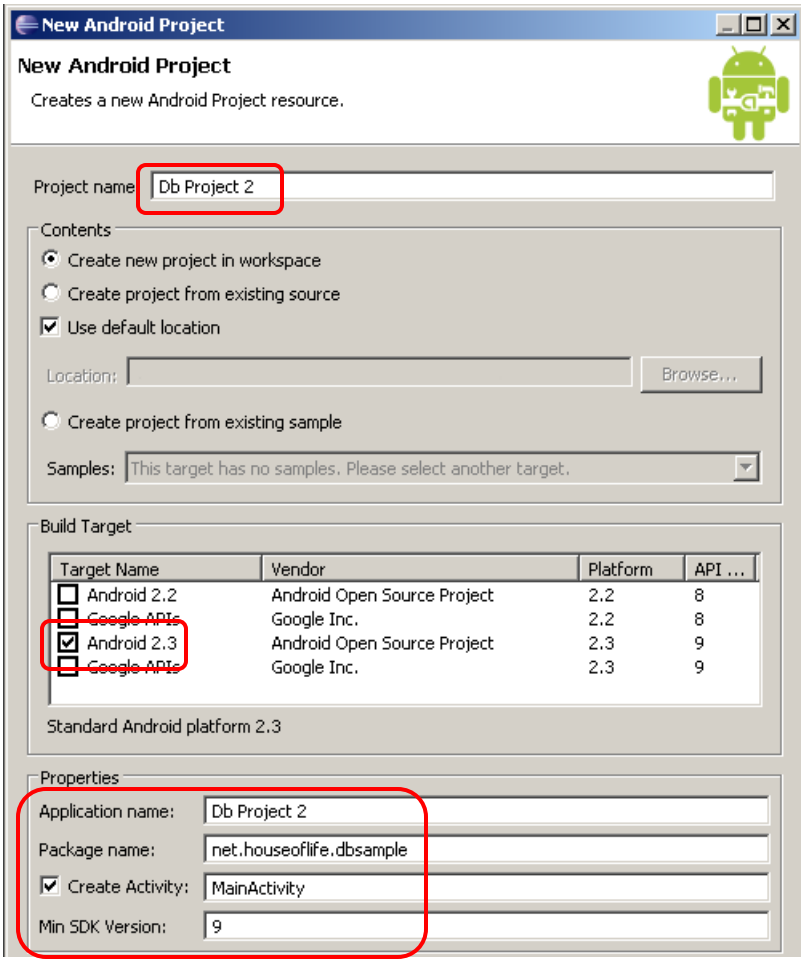
Pengguna melengkapi semua field kemudian menekan tombol Save untuk menyimpan data.



Kembali pada tampilan utama. Saat tombol Browse ditekan, data-data akan ditampilkan secara bergantian pada kotak pesan mulai dari awal hingga akhir.

Berikut ini adalah uraian mengenai pembuatan program tersebut.

1. Jalankan **Eclipse** dan buat sebuah **Android Project**.



2. Buat sebuah class baru bernama **DBAdapter** dengan kode program di bawah ini. Anda tidak perlu mengetikkan sendiri kode program itu karena program selengkapnya bisa diunduh dari situs web dimana artikel ini berasal.

   Class ini dibuat untuk menempatkan sub-program yang menangani operasi-operasi database seperti pembuatan table, pengisian data, perubahan data, penghapusan data dan pemanggilan data. Tujuannya adalah tidak mencampur-adukkan logika bisnis dengan operasi tingkat rendah (SELECT, INSERT, UPDATE dan DELETE) yang akan membuat kode program sulit dibaca dan berkesan rumit sekali. Untuk tujuan kerapihan program, akan jauh lebih baik jika program utama fokus hanya pada jalannya alur program utama, sementara fungsi-fungsi database dipisahkan ke dalam class DBAdapter ini. Program utama cukup memanggil fungsi-fungsinya secara sederhana dengan satu baris perintah saja, walaupun sebetulnya fungsi itu ditulis dengan baris yang panjang dan proses yang rumit di bagian sub-program.

```
package net.houseoflife.dbsample;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DBAdapter {
    private static final String TAG="DBAdapter";
    private static final String DATABASE_NAME="mycompany.sqlite";
    private static final int DATABASE_VERSION=1;
    private static final String TABLE_CREATE = "create table customers (_id integer primary key autoincrement, "
        + "custname text not null, custaddr text not null, "
        + "custgender text not null, custphone text not null)";
    private static final String TABLE_DROP = "DROP TABLE IF EXISTS customers";

    public static final String KEY_ROWID="_id";
    public static final String KEY_CUSTNAME="custname";
    public static final String KEY_CUSTADDR="custaddr";
    public static final String KEY_CUSTGENDER="custgender";
    public static final String KEY_CUSTPHONE="custphone";
    private final Context context;
    private DatabaseHelper dbHelper;
    private SQLiteDatabase db;

    public DBAdapter(Context ctx) {
        this.context = ctx;
        dbHelper = new DatabaseHelper(this.context);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper {
        DatabaseHelper(Context ctx) {
            super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db) {
            db.execSQL(TABLE_CREATE);
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            Log.w(TAG, "Upgrading database from version " + oldVersion + " to " + newVersion
                    + ", which will destroy all old data");
```

```
            db.execSQL(TABLE_DROP);
            onCreate(db);
        }
    }

    public DBAdapter open() throws SQLException {
        db = dbHelper.getWritableDatabase();
        return this;
    }

    public void close() {
        dbHelper.close();
    }

    public long insertCustomer(String custName, String custAddr, char custGender, String custPhone) {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_CUSTNAME, custName);
        initialValues.put(KEY_CUSTADDR, custAddr);
        initialValues.put(KEY_CUSTGENDER, Character.toString(custGender));
        initialValues.put(KEY_CUSTPHONE, custPhone);

        return db.insert("customers", null, initialValues);
    }

    public Cursor getAllCustomers() {
        return db.query("customers", new String[] {
                KEY_ROWID, KEY_CUSTNAME, KEY_CUSTADDR, KEY_CUSTGENDER, KEY_CUSTPHONE
        }, null, null, null, null, KEY_ROWID + " DESC");
    }
}
```

Dengan merujuk pada konsep *embedded database*, tampak bahwa program ini akan membuat sebuah database SQLite bernama **mycompany.sqlite** pada perangkat android yaitu *smartphone* atau *tablet* android; atau *emulator* android jika program ini dijalankan pada komputer selama masih dalam tahap pengembangan. Selanjutnya ke dalam database itu akan ditambahkan sebuah table bernama **customers** dengan perintah CREATE berikut ini,

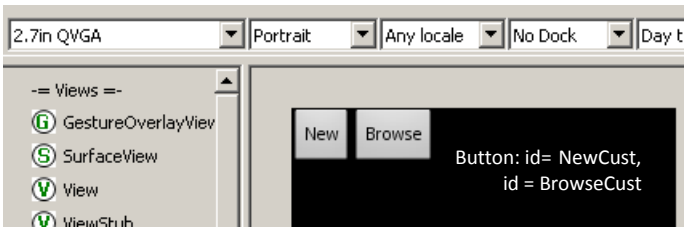
```
private static final String TABLE_CREATE = "create table customers (_id integer primary key autoincrement, "
        + "custname text not null, custaddr text not null, "
        + "custgender text not null, custphone text not null)";
```

Database dan table tersebut akan dibentuk saat program dijalankan pertama kali, yaitu melalui objek **DatabaseHelper** saat ia diinisiasi.

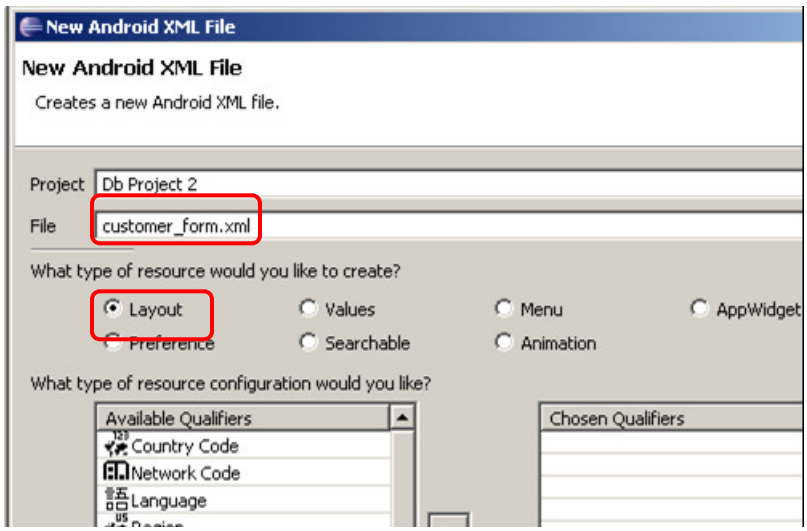
```
private static class DatabaseHelper extends SQLiteOpenHelper {
DatabaseHelper(Context ctx) {
        super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
        db.execSQL(TABLE_CREATE);
}
...
```

Pada bagian akhir dari class tersebut tampak dua buah *method* bernama **insertCustomer** dan **getAllCustomers** yang masing-masing berfungsi untuk menambahkan data ke dalam table customers dan mengambil data dari dalamnya. Perhatikan cara penggunaan fungsi **insert** dan **query** yang bisa jadi merupakan sesuatu yang baru untuk Anda, serta penggunaan objek **Cursor** untuk menampung baris-baris hasil pengambilan data.

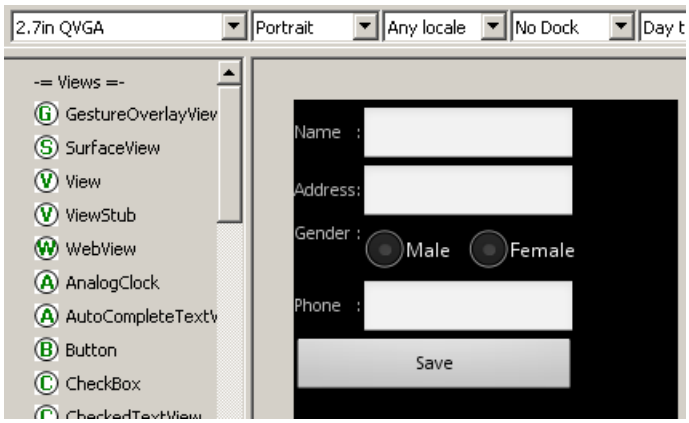
3. Buka **res → layout → main.xml** untuk merancang tampilan layar utama. Tambahkan dua buah tombol pada tampilan seperti diperlihatkan di bawah ini.



4. Buat sebuah file **Android XML** dan beri nama sebagai **customer_form.xml** dengan tipe **Layout**.



5. Silakan mendesain form pengsian data seperti diperlihatkan oleh gambar berikut ini.



EditText: id=CustName

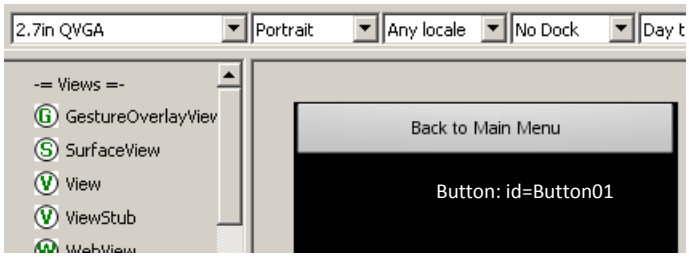
EditText: id=CustAddr

RadioButton: id=GMale, id=GFemale

EditText: id=CustPhone

Button: id=Save

6. Buat sebuah file Android XML lain dan beri nama sebagai **save_customer.xml** dengan tipe **Layout**. Tambahkan sebuah tombol pada tampilan seperti diperlihatkan di bawah ini.



7. Buka class **MainActivity.java** untuk menambahkan kode program sebagai berikut.

```
package net.houseoflife.dbsample;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btnNew = (Button) findViewById(R.id.NewCust
        btnNew.setOnClickListener(new View.OnClickListener() {
       public void onClick(View v) {
           Intent i = new Intent(MainActivity.this, CustomerForm.class);
                startActivity(i);
           }
       } );

        Button btnBrowse = (Button) findViewById(R.id.BrowseCust
        btnBrowse.setOnClickListener(new View.OnClickListener() {
           public void onClick(View v) {
               DBAdapter db = new DBAdapter(MainActivity.this);
               db.open();
               Cursor c = db.getAllCustomers();
               if (c.moveToFirst()) {
                   do {
                       Toast.makeText(MainActivity.this, c.getString(1) + ", " + c.getString(2), Toast.LENGTH_SHORT).show();
                   } while (c.moveToNext());
               }
               else
                   Toast.makeText(MainActivity.this, "No data", Toast.LENGTH_SHORT).show();
               db.close();
           }
       } );
    }
}
```

Bagian terpenting dari program ini adalah menagani aksi penekanan tombol New dan Browse. Saat tombol New ditekan maka class *CustomerForm* akan dipanggil untuk menampilkan layar yang ditunjukkan oleh customer_form.xml, yaitu untuk pengisian data.

Namun saat tombol Browse ditekan, maka data yang tersimpan di dalam database SQLite akan ditampilkan melalui baris program berikut ini:

```
...
Cursor c = db.getAllCustomers();
if (c.moveToFirst()) {
        do {
                ...
        } while (c.moveToNext());
...
```

8. Buat sebuah class baru bernama **CustomerForm** yang berisi program sebagai berikut.

```
package net.houseoflife.dbsample;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioGroup;

public class CustomerForm extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.customer_form);

        Button btnEducation = (Button) findViewById(R.id.Save);
        btnEducation.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            //validate data entries
            String custName = ((EditText) findViewById(R.id.CustName)).getText().toString().trim();
            String custAddr = ((EditText) findViewById(R.id.CustAddr)).getText().toString().trim();
            String custPhone = ((EditText) findViewById(R.id.CustPhone)).getText().toString().trim();
            char custGender = 'X';

        switch (((RadioGroup) findViewById(R.id.CustGender)).getCheckedRadioButtonId()) {
```

```
                case R.id.GMale:
                    custGender='M';
                    break;
                case R.id.GFemale:
                    custGender='F';
                    break;
                }

                Context context=CustomerForm.this;

                if(custName.equals("") || custAddr.equals("") || custPhone.equals("") || custGender == 'X') {
                     String e = "Please complete the data.";

                     new AlertDialog.Builder(context)
                     .setTitle("Invalid Data")
                     .setMessage(e)
                     .setNeutralButton("Close", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                                // TODO Auto-generated method stub
                        }
                    }).show();
                 }
                 else {
                     //If OK, then send the data to save
                     Intent i = new Intent(CustomerForm.this, SaveCustomer.class);
                     i.putExtra("CustName", custName);
                     i.putExtra("CustAddr", custAddr);
                     i.putExtra("CustPhone", custPhone);
                     i.putExtra("CustGender", custGender);
                      startActivity(i);
                 }
            }
        } );
    }
}
```

Program di atas memeriksa isian pada form untuk memastikan semua field diisi. Saat semua data telah diisi dengan benar maka ia akan mengirim isi semua field kepada class lain yang bernama *SaveCustomer* melalui baris-baris program:

```
...
Intent i = new Intent(CustomerForm.this, SaveCustomer.class);
i.putExtra("CustName", custName);
i.putExtra("CustAddr", custAddr);
i.putExtra("CustPhone", custPhone);
```

```
        i.putExtra("CustGender", custGender);
        startActivity(i);
        ...
```

9. Buat class yang terakhir bernama **SaveCustomer** yang berisi program sebagai berikut.

```
package net.houseoflife.dbsample;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class SaveCustomer extends Activity{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.save_customer);

        //get the inputed data
        Intent i = getIntent();
        Bundle b = i.getExtras();

        String custName = b.getString("CustName");
       String custAddr = b.getString("CustAddr");
       String custPhone = b.getString("CustPhone");
       char custGender = b.getChar("CustGender");

        DBAdapter db = new DBAdapter(this

        try {
           db.open();
           long id = db.insertCustomer(custName,custAddr,custGender,custPhone);
           Toast
           .makeText(this, "Data successfully saved", Toast.LENGTH_SHORT)
           .show();
        }
        catch(Exception ex) {
           Toast
           .makeText(this, "Saving error", Toast.LENGTH_SHORT)
           .show();
        }
        finally {
            db.close();
        }
```

```
        Button btnBack = (Button) findViewById(R.id.Button01);
        btnBack.setOnClickListener(new View.OnClickListener() {
           public void onClick(View v) {
            Intent i = new Intent(SaveCustomer.this, MainActivity.class);
                startActivity(i);
           }
        } );
    }
}
```

Program ini menangkap nilai-nilai field yang dikirim oleh class sebelumnya kemudian menyimpan data tersebut ke dalam table melalui fungsi insertCustomer. Operasi database dilakukan di dalam blok *try-catch* untuk mengantisipasi jika seandainya terjadi *database error* yang bisa mengakibatkan program berhenti secara mendadak. Blok try-catch akan menangkap error semacam itu secara elegan dan menampilkan pesan error tanpa mengakibatkan program berhenti.

Saat data berhasil disimpan, ia akan menampilkan pesan "Data successfully saved", dan tampak sebuah tombol yang jika ditekan akan mengembalikan tampilan ke layar utama seperti semula.

10. Pada langkah terakhir, buka file **AndroidManifest.xml** dan tambahkan/edit baris yang ditandai di bawah ini.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.houseoflife.dbsample"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="Customer Entry/View">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".CustomerForm" android:label="Customer Data"></activity>
        <activity android:name=".SaveCustomer" android:label="Save Data"></activity>
    </application>
    <uses-sdk android:minSdkVersion="7" />
</manifest>
```

11. Silakan menjalankan program tersebut pada emulator Android untuk melihat hasilnya.

Setelah Anda berhasil memasukkan beberapa data, silakan mengambil database SQLite yang ada di dalam emulator agar bisa dilihat isinya menggunakan program *SQLite Manager.* Langkah-langkah untuk mengambil database dan membaca isinya telah dijelaskan pada artikel sebelumnya seperti yang dijelaskan pada bagian awal tulisan ini.

Sebagai penutup berikut ini adalah beberapa method penting yang ada di dalam class **SQLiteDatabase** yang merupakan operasi dasar pengoperasian database SQLite. Penggunaan method tersebut harus dipahami karena mereka hampir selalu digunakan di dalam aplikasi database.

Method	**Fungsi**
insert	Menambah baris data ke dalam sebuah table
query	Mencari/mengambil data dari sebuah table
update	Mengubah data dalam sebuah table
delete	Menghapus data dalam sebuah table
execSQL	Menjalankan perintah SQL di dalam database

Di luar method tersebut masih ada banyak method lain yang perlu Anda ketahui pemakaiannya untuk mengoptimalkan penggunaan database SQLite di dalam aplikasi Android. Silakan membaca referensi di bawah ini untuk informasi lebih lengkap mengenai method dan perintah SQL:

- http://www.sqlite.org/lang.html
- http://www.sqlite.org/lang_corefunc.html
- http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html