Konsep Dasar Database SQLite pada Android

Menggunakan database SQLite adalah salah satu pilihan yang bisa kita pakai untuk menyimpan data ke dalam aplikasi Android . Karena database SQLite ini merupakan sebuah database manajemen system yang memang dirancang untuk digunakan didalam perangkat android.

Database **SQLite** adalah solusi penyimpanan yang baik jika anda memiliki data terstruktur yang perlu diakses dan disimpan secara persisten serta sering ditelusuri dan diubah. Anda juga bisa menggunakan SQLite sebagai media penyimpanan utama untuk data aplikasi atau pengguna, atau anda juga bisa menggunakannya untuk proses *caching* serta menyediakan data yang diambil dari *cloud*. Karena hanya digunakan dalam perangkat mobile yang tentu saja memiliki memory terbatas, tentu saja SQLite ini hanya memiliki kemampuan atau fitur yang terbatas saja, maka jangan anda bayangkan bahwa SQLite ini memiliki fitur seperti halnya database server sekelas SQLServer, MySQL, Oracle dan sebagainya.

Jika anda menggunakan database SQLite, yang dinyatakan sebagai objek **SQLiteDatabase** adalah semua interaksi dengan database melalui *instance* dari kelas **SQLiteOpenHelper** yang akan mengeksekusi permintaan dan pengelolaan database. Aplikasi anda hanya boleh berinteraksi dengan SQLiteOpenHelper.

Ada dua tipe data yang dikaitkan secara khusus dengan penggunaan database SQLite, yaitu **Cursor** dan **ContentValues**.

Cursor

SQLiteDatabase selalu menyajikan hasil berupa Cursor dalam format tabel yang menyerupai database SQL. Anda bisa menganggap data sebagai larik baris dan cursor adalah pointer ke dalam satu baris data terstruktur. Kelas Cursor menyediakan metode untuk menggerakan cursor melalui struktur data, dan metode untuk mendapatkan data dari bidang-bidang setiap baris.

Kelas Cursor memiliki sejumlah subkelas yang mengimplementasikan cursor untuk tipe data tertentu, yaitu:

- **SQLiteCursor** untuk mengekspos hasil *query* dari sebuah SQLiteDatabase. SQLiteCursor tidak disinkronkan secara internal, sehingga kode yang menggunakan SQLiteCursor dari beberapa *thread* harus melakukan sinkronisasi sendiri saat menggunakan SQLiteCursor.
- **MatrixCursor** adalah implementasi cursor lengkap dan tidak tetap, yang didukung oleh deretan objek yang secara otomatis meluaskan kapasitas internal bila diperlukan.

Beberapa method yang ada pada cursor adalah :

- **getCount()**, mengembalikan jumlah baris dalam cursor.
- **getColumnNames()**, mengembalikan deretan string yang berisi nama semua kolom dalam rangkaian hasil dalam urutan pencantumannya dalam sebuah *hasil*.
- **getPosition()**, mengembalikan posisi cursor saat ini dalam rangkaian baris.

- **Getter** tersedia untuk tipe data tertentu, seperti **getString(int column)** dan **getInt(int column)**.
- Operasi seperti **moveToFirst()** dan **moveToNext()** akan menggerakan cursor.
- **close()**, membebaskan semua sumber daya dan membuat cursor menjadi tidak valid. Ingat, untuk menutup panggilan, guna membebaskan sumber daya

Memproses Cursor

Cursor dimulai sebelum baris *hasil* pertama, sehingga pada pengulangan pertama arahkan atau posisikan *cursor* ke *hasil* pertama jika ada. Jika cursor kosong, atau baris terakhir sudah diproses, maka akan keluar dari *loop* (perulangan). Jangan lupa untuk menutup cursor bila anda telah selesai menggunakannya.

```
// menjalankan sebuah query dan menyimpan hasil dalam
// sebuah Cursor
Cursor cursor = db.rawQuery(...);
try {
   while (cursor.moveToNext()) {
   // Di dalam proses perulangan ini tempat pemrosesan data
   }
} finally {
   cursor.close();
}
```

Jika menggunakan database SQL, anda bisa mengimplementasikan kelas SQLiteOpenHelper untuk mengembalikan cursor ke aktivitas pemanggil atau adapter, atau anda bisa mengonversi data ke format yang lebih cocok untuk adapter.

ContentValues

Instance ContentValues menyimpan data sebagai pasangan nilai kunci, dalam hal ini kuncinya adalah *nama kolom* dan nilainya untuk *cell*. Satu instance ContentValues menyatakan satu baris tabel.

Metode **insert()** untuk database memerlukan nilai untuk mengisi baris yang diteruskan sebagai instance ContentValues.

```
ContentValues values = new ContentValues();
// Menambah 1 baris data. Untuk menambah banyak baris data,gunakan
// perulangan
values.put(KEY_WORD, "Android");// KEY_WORD adalah kunci,
                                //"Android" adalah value/data
values.put(KEY_DEFINITION, "Mobile operating system.");
db.insert(WORD_LIST_TABLE, null, values);
```

Mengimplementasikan Database SQLite

Untuk mengimplementasikan SQLite, anda perlu melakukan ini :

1. Buat model data.
2. Jadikan SQLiteOpenHelper sebagai subkelas
   - Gunakan konstanta untuk nama tabel dan query pembuatan database.
   - Implementasikan **onCreate()** untuk membuat SQLiteDatabase bersama tabel untuk data anda.
   - Implementasikan **onUpgrade()**.
   - Implementasikan metode **opsional**.
3. Implementasikan metode **query(), insert(), delete(), update(), count()** dalam SQLiteOpenHelper.
4. Dalam MainActivity anda, buat *instance* SQLiteOpenHelper.
5. Panggil metode SQLiteOpenHelper untuk digunakan bersama database anda.

Catatan:

- Bila anda mengimplementasikan metode, selalu masukkan operasi database ke dalam blok *try/catch*.
- Aplikasi contoh tidak memvalidasi data pengguna. Bila anda menulis aplikasi untuk dipublikasikan, selalu pastikan data pengguna sesuai harapan untuk menghindari injeksi perintah SQL yang berbahaya ke dalam database Anda.

Pembuatan Model Data

Praktik yang baik adalah dengan membuat kelas yang menyatakan data dengan *getter* dan *setter*. Untuk database SQLite, *instance* kelas ini dapat menyatakan satu catatan, dan untuk database sederhana, satu baris dalam tabel.

```
public class WordItem {
   private int mId;
   private String mWord;
   private String mDefinition;
   // Getters and setters and more
}
```

Jadikan SQLiteOpenHelper sebagai subkelas

Open helper apa pun yang Anda buat harus meng-*extends SQLiteOpenHelper*.

```
public class WordListOpenHelper extends SQLiteOpenHelper {
   public WordListOpenHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
      Log.d(TAG, "Construct WordListOpenHelper");
   }
}
```

Definisikan konstanta untuk nama tabel

Walaupun tidak diwajibkan, biasakan untuk mendeklarasikan nama tabel, kolom, dan baris sebagai konstanta. Hal ini akan membuat lebih mudah saat mengubah nama, dan *query* akan terlihat lebih mirip dengan SQL. Anda bisa melakukannya dalam kelas open helper, atau dalam kelas publik tersendiri.

```
private static final int DATABASE_VERSION = 1;
private static final String WORD_LIST_TABLE = "word_entries";
private static final String DATABASE_NAME = "wordlist";
// Nama-nama kolom...
public static final String KEY_ID = "_id";
public static final String KEY_WORD = "word";
// ... dan sebuah string array dari kolom-kolom tersebut
private static final String[] COLUMNS = {KEY_ID, KEY_WORD};
```

Definisikan query untuk membuat database

Anda memerlukan *query* yang membuat tabel untuk membuat database. Ini biasanya juga didefinisikan sebagai konstanta string. Contoh dasar ini membuat satu tabel dengan satu kolom untuk ID bertambah-otomatis/auto increment dan kolom untuk menampung kata.

```
private static final String WORD_LIST_TABLE_CREATE =
        "CREATE TABLE " + WORD_LIST_TABLE + " (" +
         KEY_ID + " INTEGER PRIMARY KEY, " +
         // auto-increment
         KEY_WORD + " TEXT );";
```

Implementasikan onCreate() dan buat database

Metode **onCreate()** hanya dipanggil jika tidak ada database. Buat tabel Anda dalam metode, dan boleh menambahkan data awal.

```
@Override
public void onCreate(SQLiteDatabase db) { // Creates new database
   db.execSQL(WORD_LIST_TABLE_CREATE); // Create the tables
   fillDatabaseWithData(db); // Add initial data

}
```

Implementasikan onUpgrade()

Ini adalah metode yang diperlukan. Jika database hanya berfungsi sebagai cache untuk data yang juga disimpan online, Anda bisa menghapus tabel dan membuat ulang setelah peningkatan versi selesai.

Catatan: Jika database adalah *storage* utama, Anda harus mengamankan data pengguna sebelum melakukannya karena operasi ini akan memusnahkan semua data.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
   // Pertama-tama kita harus menyimpan data User lebih dahulu
   Log.w(WordListOpenHelper.class.getName(),
         "Upgrading database from version " + oldVersion + " to "
          + newVersion + ", which will destroy all old data");
   db.execSQL("DROP TABLE IF EXISTS " + WORD_LIST_TABLE);
   onCreate(db);
}
```

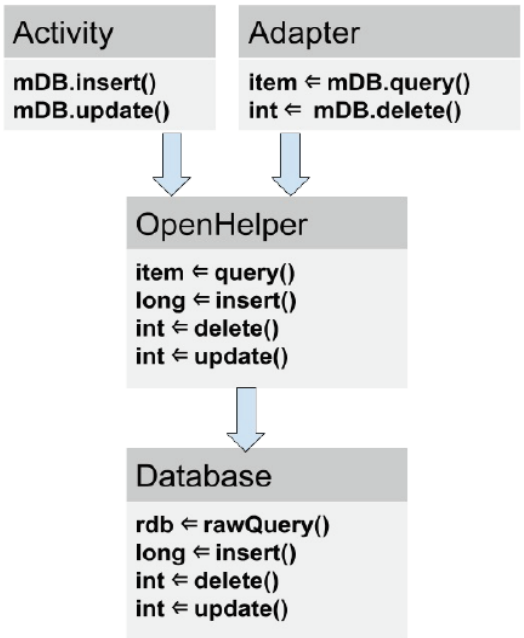
Metode opsional

Kelas open helper menyediakan metode tambahan yang bisa Anda ganti bila diperlukan.

- **onDowngrade()**, Implementasi default menolak penurunan versi.
- **onConfigure()**, dipanggil sebelum onCreate. Gunakan ini hanya untuk memanggil metode yang mengonfigurasi parameter koneksi database.
- **onOpen(),** pekerjaan apa pun selain konfigurasi yang harus dilakukan sebelum database dibuka.

Operasi Database

Walaupun bisa memanggil metode dalam open helper yang diinginkan dan mengembalikan yang anda pilih ke aktivitas pemanggil, lebih baik lanjutkan dengan metode **query(), insert(), delete(), update(), count()** standar yang sesuai dengan API database dan penyedia materi/*content provider*. Diagram berikut menampilkan cara mendesain API yang berbeda agar konsisten dan jelas.



query()

Metode kueri yang diimplementasikan dalam kelas open helper bisa mengambil dan mengembalikan tipe data apa pun yang diperlukan antarmuka pengguna. Karena open helper menyediakan metode praktis untuk menyisipkan, menghapus, dan memperbarui baris, metode query tidak perlu generik .

Database menyediakan dua metode untuk mengirimkan query: ***SQLiteDatabase.rawQuery()*** dan ***SQLiteDatabase.query()***, bersama sejumlah pilihan untuk argumen/parameter.

SQLiteDatabase.rawQuery()

Metode kueri open helper bisa membentuk kueri SQL dan mengirimkannya sebagai rawQuery ke database yang mengembalikan kursor. Jika data disediakan oleh aplikasi dan dikontrol penuh, kita bisa menggunakan rawQuery().

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

- Parameter pertama untuk db.rawQuery() adalah string query SQLite.
- Parameter kedua berisi argumen.

```
cursor = mReadableDB.rawQuery(queryString, selectionArgs);
```

SQLiteDatabase.query()

Jika Anda memproses data yang disediakan pengguna, bahkan setelah validasi, lebih aman membentuk query dan menggunakan versi metode SQLiteDatabase.query(). Argumen di sini adalah yang diharapkan dalam SQL dan didokumentasikan dalam dokumentasi SQLiteDatabase.

```
Cursor query (boolean distinct, String table, String[] columns,
              String selection, String[] selectionArgs, String
              groupBy, String having, String orderBy,String limit)
```

Seperti ini contoh dasarnya :

```
String[] columns = new String[]{KEY_WORD};
String where = KEY_WORD + " LIKE ?";
searchString = "%" + searchString + "%";
String[] whereArgs = new String[]{searchString};
cursor = mReadableDB.query(WORD_LIST_TABLE, columns, where, whereArgs,
null, null, null);
```

Contoh lengkap open helper query()

```
public WordItem query(int position) {
   String query = "SELECT * FROM " + WORD_LIST_TABLE +
                  " ORDER BY " + KEY_WORD + " ASC " +
                  "LIMIT " + position + ",1";
   Cursor cursor = null;
   WordItem entry = new WordItem();
   try {
      if (mReadableDB == null) {mReadableDB = getReadableDatabase();}
      cursor = mReadableDB.rawQuery(query, null);
      cursor.moveToFirst();
      entry.setId(cursor.getInt(cursor.getColumnIndex(KEY_ID)));
      entry.setWord(cursor.getString(cursor
                    .getColumnIndex(KEY_WORD)));
   } catch (Exception e) {
      Log.d(TAG, "EXCEPTION! " + e);
   } finally {
      // Must close cursor and db now that we are done with it.
      cursor.close();
      return entry;
   }
}
```

insert()

Metode insert() open helper memanggil SQLiteDatabase.insert(), yaitu metode SQLiteDatabase praktis yang digunakan untuk menyisipkan baris atau menambahkan baris data ke dalam database.

```
long insert(String table, String nullColumnHack, ContentValues values)
```

- Argumen pertama adalah nama tabel.
- Argumen kedua adalah sebuah String nullColumnHack . Ini adalah solusi yang memungkinkan kita untuk menyisipkan baris kosong. Lihat dokumentasi untuk insert(). Gunakan null.
- Argumen ketiga harus berupa kontainer *[ContentValues]* bersama nilai-nilai untuk mengisi baris. Contoh ini hanya memiliki satu kolom; untuk tabel dengan banyak kolom, tambahkan
- Metode database mengembalikan ID item yang baru disisipkan, dan Anda harus meneruskannya ke aplikasi.

Contoh :

```
newId = mWritableDB.insert(WORD_LIST_TABLE, null, values);
```

delete()

Metode delete dari open helper memanggil metode delete() database, yang praktis digunakan untuk menghapus data sehingga anda tidak perlu menulis query SQL seluruhnya.

```
int delete (String table, String whereClause, String[] whereArgs)
```

- Argumen pertama adalah nama tabel.
- Argumen kedua adalah sebuah klausa WHERE.
- Argumen ketiga adalah argumen untuk klausa WHERE.

Anda bisa menghapus menggunakan kriteria apa pun, dan metode akan mengembalikan jumlah item yang sebenarnya dihapus, yang juga harus dikembalikan oleh open helper.

Contoh:

```
deleted = mWritableDB.delete(WORD_LIST_TABLE,
                KEY_ID + " =? ", new String[]{String.valueOf(id)});
```

update()

Metode update dari open helper memanggil metode update() database, yang praktis digunakan sehingga Anda tidak perlu menulis query SQL seluruhnya. Argumen tersebut sudah familiar dari metode sebelumnya, dan onUpdate mengembalikan jumlah baris yang diperbarui.

```
int update(String table, ContentValues values,
           String whereClause, String[] whereArgs)
```

- Argumen pertama adalah nama tabel.
- Argumen kedua harus berupa ContentValues bersama nilai-nilai baru untuk baris tersebut.
- Argumen ketiga adalah klausa WHERE.
- Argumen keempat adalah argumen untuk klausa WHERE.

Contoh :

```
ContentValues values = new ContentValues();
values.put(KEY_WORD, word);
mNumberOfRowsUpdated = mWritableDB.update(WORD_LIST_TABLE,
values, // new values to insert
KEY_ID + " = ?",
new String[]{String.valueOf(id)});
```

count()

Metode count() mengembalikan jumlah entri dalam database. Jika menggunakan RecyclerView.Adapter, Anda harus mengimplementasikan getItemCount(), yang perlu mendapatkan sejumlah baris dari open helper, yang perlu didapat dari database.

Contoh kode yang bisa di impelemtasikan di adapter untuk tampilan RecyclerView :

```
@Override
public int getItemCount() {
   return (int) mDB.count();
}
```

Di open helper :

```
public long count(){
if (mReadableDB == null) {mReadableDB = getReadableDatabase();}
   return DatabaseUtils.queryNumEntries(mReadableDB,
          WORD_LIST_TABLE);
}
```

queryNumEntries() adalah metode di kelas DatabaseUtils publik, yang menyediakan banyak metode praktis untuk digunakan bersama kursor, database, dan juga penyedia materi/content provider.

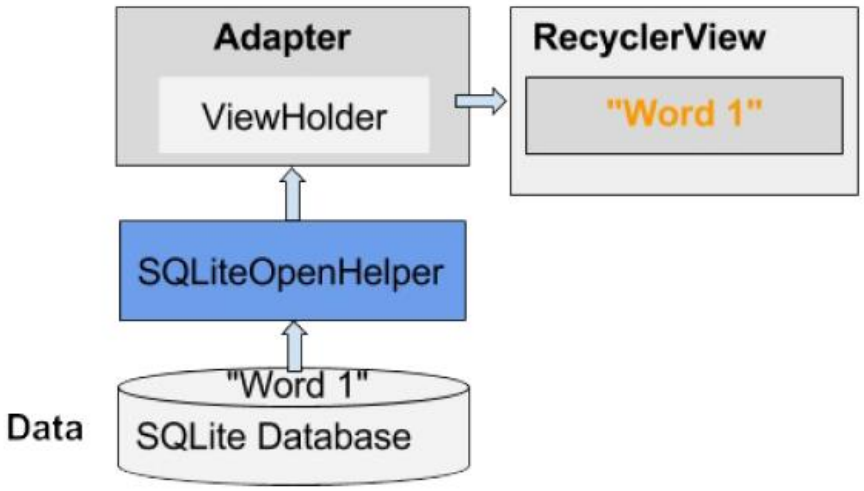
Buat Instance Open Helper

Untuk menangani database, dalam MainActivity, di onCreate, panggil :

```
mDB = new WordListOpenHelper(this);
```

Bekerja dengan Database

Mengombinasikan backend SQLiteDatabase dengan RecyclerView untuk menampilkan data merupakan pola yang umum.



Misalnya :

- Menekan FAB (*Float Action Button*) bisa memulai aktivitas/activity yang akan mendapatkan masukan dari pengguna dan menyimpannya ke dalam database sebagai item baru atau diperbarui.
- Menggesek suatu item bisa menghapusnya setelah pengguna mengonfirmasi penghapusan.

Transaksi

Gunakanlah transaksi jika:

- Saat melakukan beberapa operasi yang semuanya harus diselesaikan agar database konsisten, misalnya, memperbarui penetapan harga item terkait untuk kejadian penjualan.
- Untuk *batch* beberapa operasi independen guna meningkatkan kinerja, seperti penyisipan banyak data.

Transaksi bisa disarangkan, dan kelas SQLiteDatabase menyediakan metode tambahan untuk mengelola transaksi tersarang. Lihat dokumentasi referensi SQLiteDatabase.

Idiom Transaksi

```
db.beginTransaction();
try {
   ...
   db.setTransactionSuccessful();
} finally {
   db.endTransaction();
}
```

Mengirim Database bersama aplikasi anda

Kadang-kadang Anda mungkin ingin menyertakan database yang sudah terisi bersama aplikasi. Ada sejumlah cara untuk melakukannya, dan konsekuensi dari setiap cara tersebut:

- Sertakan perintah SQL bersama aplikasi dan perintahkan untuk membuat database dan menyisipkan data pada penggunaan pertama. Inilah yang pada dasarnya akan Anda lakukan dalam praktik menyimpan data. Jika jumlah data yang ingin dimasukkan dalam database kecil, hanya satu contoh agar pengguna bisa melihat sesuatu, Anda bisa menggunakan metode ini.
- Kirimkan data bersama APK sebagai sumber daya, dan bangun database saat pengguna membuka aplikasi untuk pertama kali. Metode ini sama dengan metode pertama, namun sebagai ganti mendefinisikan data dalam kode, Anda memasukkannya dalam sumber daya, misalnya, dalam format CSV. Selanjutnya Anda bisa membaca data dengan aliran masukan dan menambahkannya ke database.
- Bangun dan isi dahulu database SQLite lalu sertakan dalam APK. Dengan metode ini, tulis aplikasi yang akan membuat dan mengisi database. Anda bisa melakukan ini pada emulator. Selanjutnya Anda bisa menyalin file tersebut di tempat penyimpanan database sebenarnya (direktori “/data/data/YOUR_PACKAGE/databases/”) dan menyertakannya sebagai aset bersama aplikasi. Saat aplikasi dimulai untuk pertama kali, salin kembali file database ke dalam direktori “/data/data/YOUR_PACKAGE/databases/”.

PROJECT

Kita akan membuat sebuah aplikasi sederhana untuk mengelola data mahasiswa dalam sebuah table yang berisi kolom-kolom id, nama, dan nomor telepon.

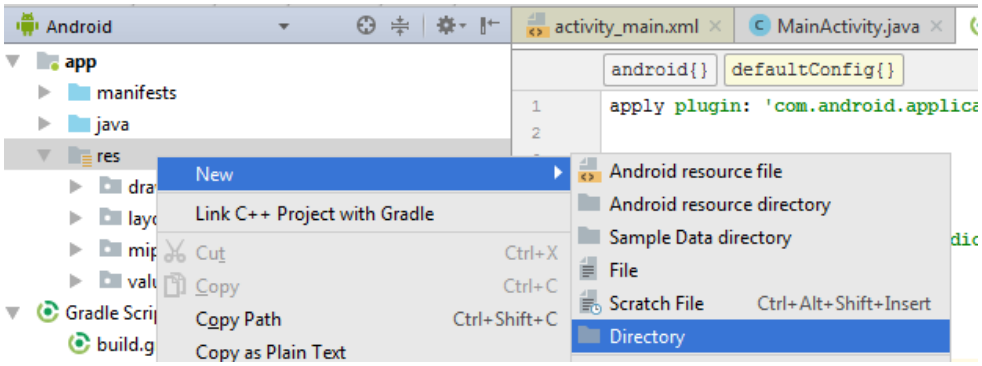
- Buatlah project baru dengan nama DataMahasiswa
- Expand folder res/values. Edit file string.xml dengan source berikut ini

```
<resources>
    <string name="namaApp">Mahasiswa</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="Add_New">Tambah Data</string>
    <string name="edit">Edit Mahasiswa</string>
    <string name="delete">Delete Mahasiswa</string>
    <string name="title_activity_display_contact">Data
Mahasiswa</string>
    <string name="name">Name</string>
    <string name="phone">Phone</string>
    <string name="nim">No Mhs</string>
    <string name="save">Simpan</string>
    <string name="deleteContact">Yakin...ingin menghapus ini.</string>
    <string name="yes">Ya</string>
    <string name="no">Tidak</string>
    <string name="db">Data</string>
    <string name="ti">Mahasiswa TI</string>
</resources>
```

- **Buat Folder Menu**

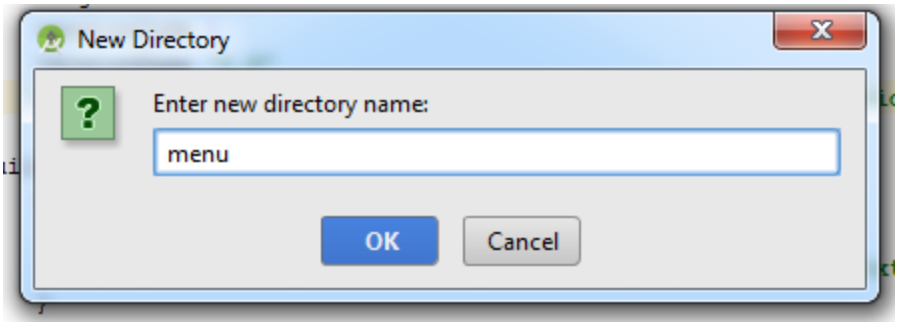
Langkah kedua buat folder / direktori dengan nama **menu** pada folder **res. Klik kanan folder res > New > Directory.**

Langkah ini dilakukan apabila di susunan folder project kita belum ada directory menu.



Membuat direktori baru

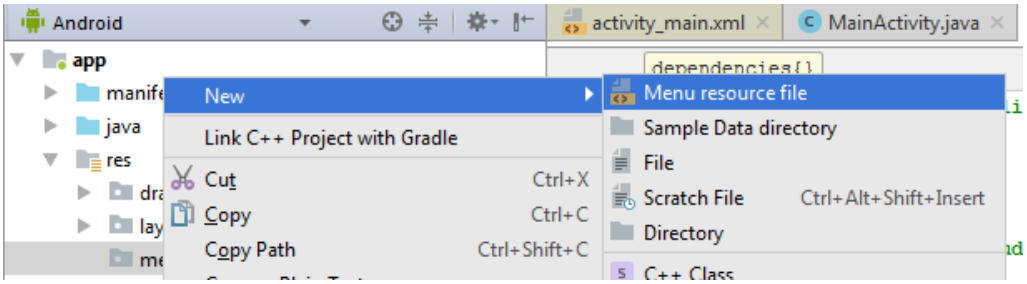
Beri nama folder **menu.**



Memberi nama direktori

- **Buat Layout menu_main.xml dan menu_display.xml**

Klik kanan pada direktori **menu** yang sudah dibuat pada point 2 > pilih **menu resource file**



Membuat layout menu_main

File menu_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item android:id="@+id/item1"
    android:title="@string/Add_New" >
</item>
</menu>
```

File menu_display.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/Edit_Contact"
        android:orderInCategory="100"
        android:title="@string/edit"/>

    <item
        android:id="@+id/Delete_Contact"
        android:orderInCategory="100"
        android:title="@string/delete"/>
</menu>
```

- Buat kelas java sebagai tempat mendefinisikan dan memanipulasi database. Jadi kelas ini adalah sebuah klas java (bukan activity) yang merupakan turunan dari klas ***SQLiteOpenHelper***. Keberadaan klas bantu ini adalah wajib atau mutlak harus ada, jika membuat sebuah aplikasi android yang menggunakan database SQLite untuk menyimpan dan mengelola data-data kita.
  Klik kanan folder java > (package an) > New > Java Class ,lalu beri nama DBHelper.
  File DBHelper :

```
package com.example.datamahasiswa;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Hashtable;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase;

public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "MyDBName.db";
    public static final String MHS_TABLE_NAME = "mahasiswa";
    public static final String MHS_COLUMN_ID = "id";
    public static final String MHS_COLUMN_NIM = "nim";
    public static final String MHS_COLUMN_NAMA = "nama";
    public static final String MHS_COLUMN_PHONE = "phone";
    private HashMap hp;

    public DBHelper(Context context) {
        super(context, DATABASE_NAME , null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // TODO Auto-generated method stub
        db.execSQL(
                "create table mahasiswa " +
                        "(id integer primary key, nim text,nama
text,phone text)"
        );
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        // TODO Auto-generated method stub
        db.execSQL("DROP TABLE IF EXISTS mahasiswa");
        onCreate(db);
    }

    public boolean insertContact (String nim, String nama, String
phone) {
        SQLiteDatabase db = this.getWritableDatabase();
```

```
        ContentValues contentValues = new ContentValues();
        contentValues.put("nim", nim);
        contentValues.put("nama", nama);
        contentValues.put("phone", phone);

        db.insert("mahasiswa", null, contentValues);
        return true;
    }

    public Cursor getData(int id) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res =  db.rawQuery( "select * from mahasiswa where
id="+id+"", null );
        return res;
    }

    public int numberOfRows(){
        SQLiteDatabase db = this.getReadableDatabase();
        int numRows = (int) DatabaseUtils.queryNumEntries(db,
MHS_TABLE_NAME);
        return numRows;
    }


    public ArrayList<String> getAllCotacts() {
        ArrayList<String> array_list = new ArrayList<String>();

        //hp = new HashMap();
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res =  db.rawQuery( "select * from mahasiswa", null );
        res.moveToFirst();

        while(res.isAfterLast() == false){

array_list.add(res.getString(res.getColumnIndex(MHS_COLUMN_NAMA)));
            res.moveToNext();
        }
        return array_list;
    }
}
```

- Buatlah sebuah activity baru, beri nama DisplayMhs.
  Caranya : **Klik kanan folder java > (package an) > New > Activity > Empty Activity**
- Edit layout utama *activity_main.xml* berikut :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:id="@+id/textView"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:textSize="30sp"
            android:text="@string/db"
            android:textColor="#aa55ff" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/ti"
            android:id="@+id/textView2"
            android:layout_below="@+id/textView"
            android:layout_centerHorizontal="true"
            android:textSize="35sp"
            android:textColor="#aa55ff" />

            <ListView
                android:id="@+id/listView1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/textView2"
                android:layout_centerHorizontal="true"
                android:layout_centerVertical="true" >
            </ListView>
    </RelativeLayout>
```

- Edit layout display *activity_display_mhs.xml* berikut :

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context=".DisplayMhs">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="370dp"
        android:paddingLeft="10sp"
        android:paddingTop="10sp"
        android:paddingRight="100sp"
        android:paddingBottom="100sp">

        <EditText
            android:id="@+id/editTextNim"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_marginLeft="82dp"
            android:layout_marginTop="5dp"
            android:ems="10"
            android:inputType="text"></EditText>
```

```
<EditText
    android:id="@+id/editTextPhone"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editTextName"
    android:layout_alignLeft="@+id/editTextName"
    android:layout_marginLeft="-5dp"
    android:layout_marginTop="40dp"
    android:ems="10"
    android:inputType="phone|text" />

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/editTextName"
    android:layout_alignParentLeft="true"
    android:layout_marginLeft="12dp"
    android:layout_marginBottom="91dp"
    android:text="@string/nim"
    android:textAppearance="?android:attr/textAppearanceMedium" />

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editTextPhone"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="137dp"
    android:layout_marginLeft="137dp"
    android:layout_marginTop="60dp"
    android:layout_marginBottom="49dp"
    android:onClick="run"
    android:text="@string/save" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/textView1"
    android:layout_alignBottom="@+id/editTextPhone"
    android:layout_marginLeft="10dp"
    android:layout_marginBottom="14dp"
    android:text="@string/phone"
    android:textAppearance="?android:attr/textAppearanceMedium" />

<TextView
    android:id="@+id/textView5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/textView1"
    android:layout_alignBottom="@+id/editTextPhone"
    android:layout_marginLeft="10dp"
    android:layout_marginBottom="94dp"
```

```
            android:text="@string/name"
            android:textAppearance="?android:attr/textAppearanceMedium" />


        <EditText
            android:id="@+id/editTextName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/editTextNim"
            android:layout_alignLeft="@+id/editTextNim"
            android:layout_marginLeft="9dp"
            android:layout_marginTop="35dp"
            android:ems="10"
            android:inputType="text" />

    </RelativeLayout>
</ScrollView>
```

- Edit file *DisplayMhs.java* berikut :

```
package com.example.datamahasiswa;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class DisplayMhs extends AppCompatActivity {
    int from_Where_I_Am_Coming = 0;
    private DBHelper mydb ;

    TextView nomhs ;
    TextView phone;
    TextView nama;

    int id_To_Update = 0;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_mhs);
        nomhs = (TextView) findViewById(R.id.editTextNim);
        nama = (TextView) findViewById(R.id.editTextName);
        phone = (TextView) findViewById(R.id.editTextPhone);

        mydb = new DBHelper(this);
```

```
        Bundle extras = getIntent().getExtras();
        if(extras !=null) {
            int Value = extras.getInt("id");

            if(Value>0){
                //means this is the view part not the add contact part.
                Cursor rs = mydb.getData(Value);
                id_To_Update = Value;
                rs.moveToFirst();

                String no =
rs.getString(rs.getColumnIndex(DBHelper.MHS_COLUMN_NIM));
                String nam =
rs.getString(rs.getColumnIndex(DBHelper.MHS_COLUMN_NAMA));
                String phon =
rs.getString(rs.getColumnIndex(DBHelper.MHS_COLUMN_PHONE));
                if (!rs.isClosed())  {
                    rs.close();
                }
                Button b = (Button)findViewById(R.id.button1);
                b.setVisibility(View.INVISIBLE);

                nomhs.setText((CharSequence)no);
                nomhs.setFocusable(false);
                nomhs.setClickable(false);

                nama.setText((CharSequence)nam);
                nama.setFocusable(false);
                nama.setClickable(false);

                phone.setText((CharSequence)phon);
                phone.setFocusable(false);
                phone.setClickable(false);

            }
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        Bundle extras = getIntent().getExtras();

        if(extras !=null) {
            int Value = extras.getInt("id");
            if(Value>0){
                getMenuInflater().inflate(R.menu.menu_display,menu);
            } else{
                getMenuInflater().inflate(R.menu.menu_main,menu);
            }
        }
        return true;
    }

}
```

- Edit File *MainActivity.java* berikut :

```
package com.example.datamahasiswa;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Adapter;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {
    public final static String EXTRA_MESSAGE = "MESSAGE";
    private ListView obj;
    DBHelper mydb;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mydb = new DBHelper(this);
        ArrayList array_list = mydb.getAllCotacts();
        ArrayAdapter arrayAdapter=new
ArrayAdapter(this,android.R.layout.simple_list_item_1, array_list);

        obj = (ListView)findViewById(R.id.listView1);
        obj.setAdapter(arrayAdapter);
        obj.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int
arg2, long arg3) {
                // TODO Auto-generated method stub
                int id_To_Search = arg2 + 1;

                Bundle dataBundle = new Bundle();
                dataBundle.putInt("id", id_To_Search);

                Intent intent = new
Intent(getApplicationContext(),DisplayMhs.class);

                intent.putExtras(dataBundle);
                startActivity(intent);
            }
        });
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item){
        super.onOptionsItemSelected(item);

        switch(item.getItemId()) {
            case R.id.item1:Bundle dataBundle = new Bundle();
                dataBundle.putInt("id", 0);

                Intent intent = new
Intent(getApplicationContext(),DisplayMhs.class);
                intent.putExtras(dataBundle);

                startActivity(intent);
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    public boolean onKeyDown(int keycode, KeyEvent event) {
        if (keycode == KeyEvent.KEYCODE_BACK) {
            moveTaskToBack(true);
        }
        return super.onKeyDown(keycode, event);
    }
}
```