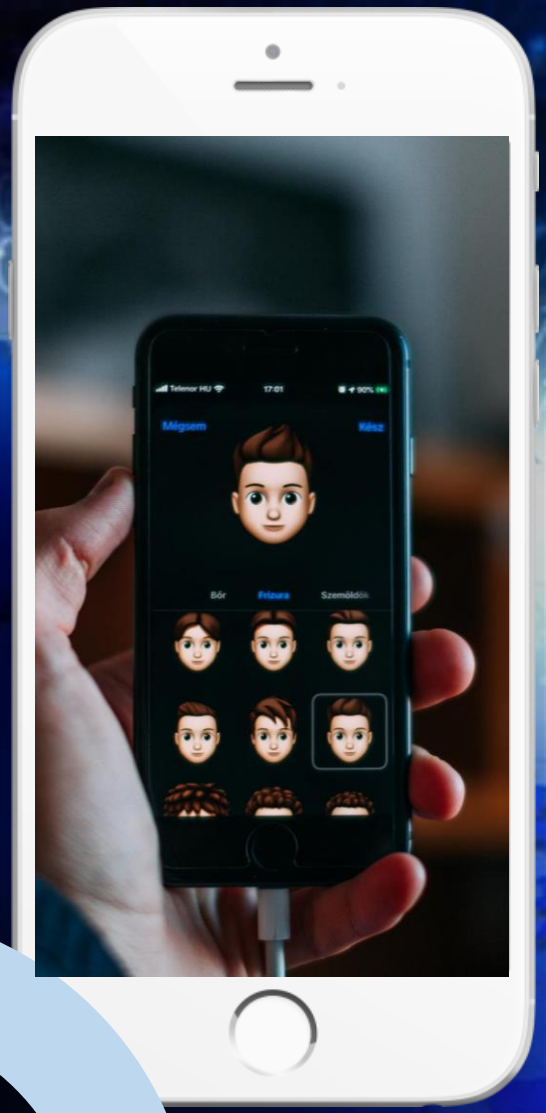


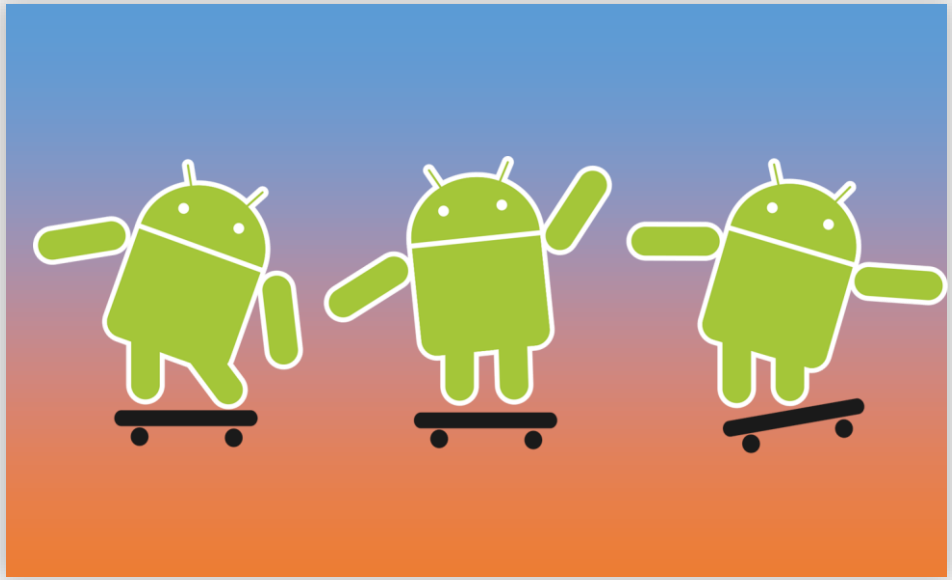
Aplikasi Database CRUD (Create, Read, Update, & Delete) menggunakan SQLite sebagai Database

Presented By Putri Hayati, S.ST, M.Kom

Kategori

9.0 SQLite Database

9.1 Operasi Database (CRUD)



SQLite Database

SQLite Database

- **Database SQLite** adalah solusi storage yang baik jika Anda memiliki data terstruktur yang perlu diakses dan disimpan secara persisten serta sering ditelusuri dan diubah.
- Anda bisa menggunakan database sebagai storage utama untuk data aplikasi atau pengguna, atau Anda bisa menggunakannya untuk meng-cache serta menyediakan data yang diambil dari awan.
- Jika Anda bisa menyatakan data berupa baris dan kolom, pertimbangkan **database SQLite.**
- Penyedia materi, yang akan diperkenalkan dalam bab berikutnya, bekerja dengan bagus pada **database SQLite**.an fungsionalitas insert, delete, update, dan **query** melalui **helper** terbuka.
- Menggunakan adaptor dan handler klik khusus untuk memungkinkan pengguna berinteraksi dengan database dari antarmuka pengguna.

SQLite Database

SQLite Database selalu menyajikan hasil berupa **Cursor** dalam format tabel yang menyerupai database SQL.

- SQLiteCursor 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 kursor lengkap dan tidak tetap, yang didukung oleh larik objek yang secara otomatis meluaskan kapasitas internal bila diperlukan.

SQLite Database

Beberapa operasi yang umum pada kursor adalah :

- getCount() mengembalikan jumlah baris dalam kursor
- getColumnNames() mengembalikan larik string yang berisi nama semua kolom dalam rangkaian hasil dalam urutan pencantumannya dalam hasil
- getPosition() mengembalikan posisi kursor 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 menggerakkan kursor
- close() membebaskan semua sumber daya dan membuat kursor menjadi tidak valid sama sekali. Ingat untuk menutup panggilan guna membebaskan sumber daya!

ContentValues

Serupa dengan cara ekstra menyimpan data, instance ContentValues menyimpan data sebagai pasangan nilai-kunci, dalam ini kuncinya adalah nama kolom dan nilainya adalah nilai untuk sel. 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();
// Insert one row. Use a loop to insert multiple rows.
values.put(KEY_WORD, "Android");
values.put(KEY_DEFINITION, "Mobile operating system.");

db.insert(WORD_LIST_TABLE, null, values);
```

Implementasi ke Database

Untuk mengimplementasikan database aplikasi Android, Anda perlu melakukan yang berikut ini :

1. (Disarankan) 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.

Model Data

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
}
```

SQLiteOpenHelper sebagai subkelas

Open helper apa pun yang Anda buat harus memperluas 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, sudah biasa mendeklarasikan nama tabel, kolom, dan baris sebagai konstanta. Hal ini akan membuat kode lebih terbaca, lebih mudah mengubahi nama, dan Query akan terlihat lebih mirip SQL. Anda bisa melakukannya dalam kelas open helper, atau dalam kelas publik tersendiri.

```
private static final int DATABASE_VERSION = 1;
 // has to be 1 first time or app will crash
 private static final String WORD_LIST_TABLE = "word_entries";
 private static final String DATABASE_NAME = "wordlist";

 // Column names...
 public static final String KEY_ID = "_id";
 public static final String KEY_WORD = "word";

 // ... and a string array of columns.
 private static final String[] COLUMNS = {KEY_ID, KEY_WORD};
```

Query untuk membuat database

Contoh dasar ini membuat satu tabel dengan satu kolom untuk ID bertambah-otomatis dan kolom untuk menampung kata.

```
private static final String WORD_LIST_TABLE_CREATE =
            "CREATE TABLE " + WORD_LIST_TABLE + " (" +
             KEY_ID + " INTEGER PRIMARY KEY, " +
             // will auto-increment if no value passed
             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
    // Cannot initialize mWritableDB and mReadableDB here,
because
    // this creates an infinite loop of on Create()
    // being repeatedly called.
}
```

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.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // SAVE USER DATA FIRST!!!
        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);
}
```

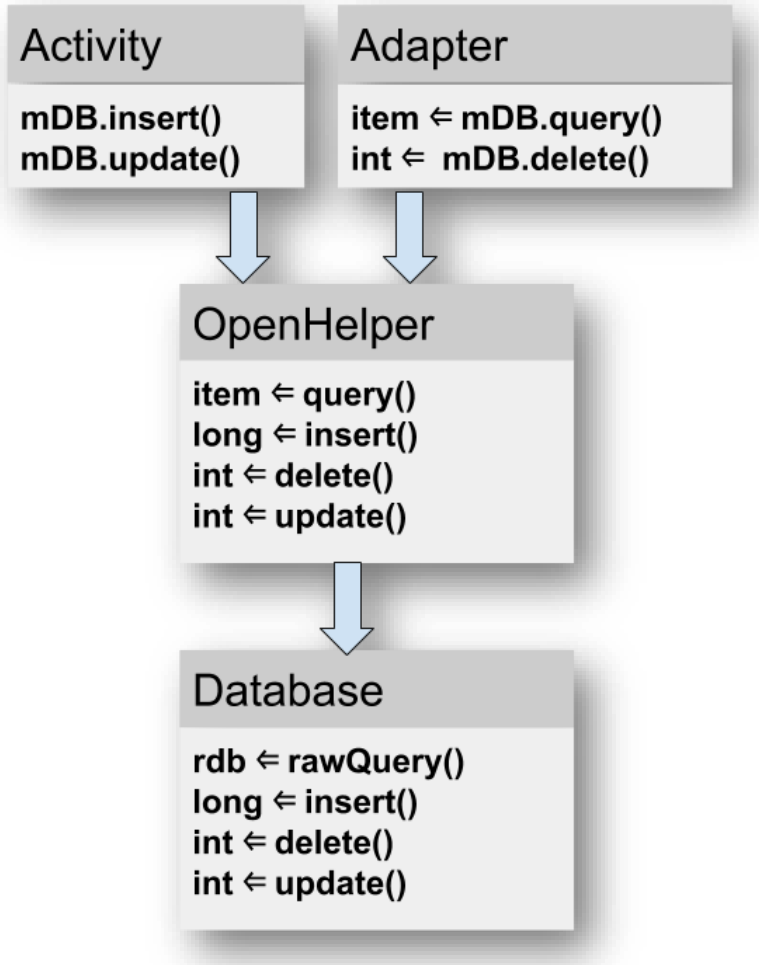
Metode Opsional

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
Create, Read, Update, Delete (CRUD)

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

Query ()

- Metode **Query** yang diimplementasikan dalam kelas **open helper** Anda bisa mengambil dan mengembalikan tipe data apa pun yang diperlukan antarmuka pengguna.
- Karena open helper menyediakan metode praktir untuk menyisipkan, menghapus, dan memperbarui baris, metode Query Anda tidak perlu generik dan mendukung operasi ini.
- Secara umum, metode **Query** Anda hanya boleh mengizinkan Query yang diperlukan oleh aplikasi dan bukan untuk keperluan umum.
- Database menyediakan dua metode untuk mengirimkan Query **SQLiteDatabase.rawQuery()** dan **SQLiteDatabase.query(),** bersama sejumlah opsi untuk argumen.

Query ()

SQLiteDatabase.rawQuery().rawQuery()

Metode Query open helper bisa membentuk Query SQL dan mengirimkannya sebagai **rawQuery** ke database yang mengembalikan kursor. Jika data disediakan oleh aplikasi dan dikontrol penuh, Anda 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);

Query ()

SQLiteDatabase.query()

Jika Anda memproses data yang disediakan pengguna, bahkan setelah validasi, lebih aman membentuk Query dan menggunakan versi metode SQLiteDatabase.query() untuk database. Argumen adalah apa yang Anda harapkan 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)
```

Inilah 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);
```

Query()

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 ke dalam database. (Metode ini praktis, karena Anda tidak perlu menulis Query SQL sendiri.)

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

- Argumen pertama adalah nama tabel.
- Argumen kedua adalah sebuah String nullColumnHack. Ini adalah solusi yang memungkinkan Anda 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 sehingga Anda tidak perlu menulis **Query SQL** seluruhnya.

```
Format :
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 familier dari metode sebelumnya, dan onUpdate mengembalikan jumlah baris yang diperbarui.

```
Format :
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.

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

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

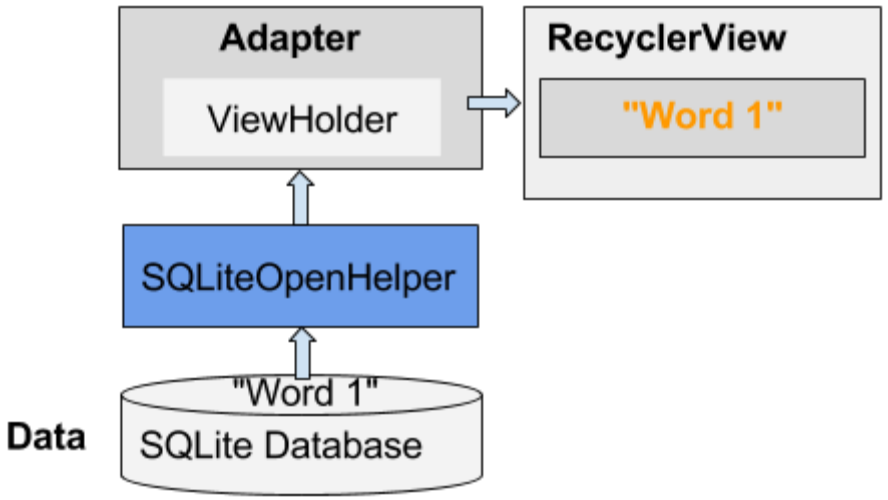
```
Di open helper :

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

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.

Any Question ?