Rakamin Academy

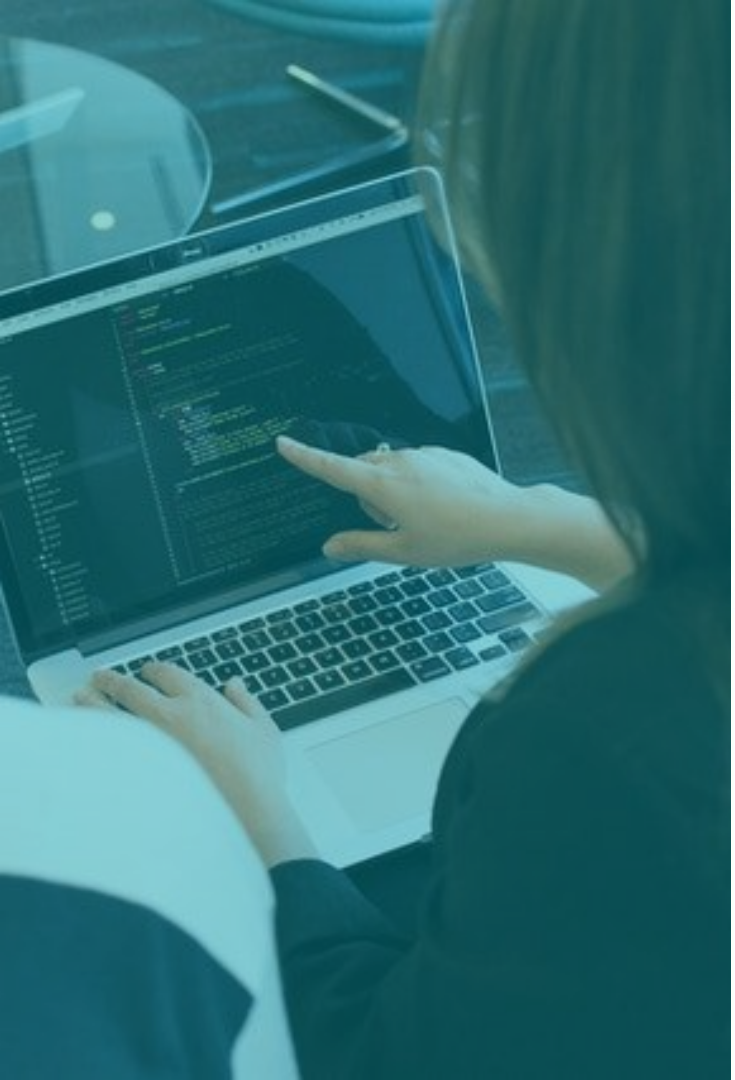
Virtual Internship Experience

mandiri

Android : SQLite Database

Disclaimer

“Dokumen ini memiliki hak cipta. Barang siapa yang menyebarluaskan atau menduplikasi tanpa izin dari instansi terkait dapat diproses sesuai dengan ketentuan hukum yang berlaku.”



Outline

1. SQLite Database
2. Komponen SQLite Database
3. Case Study

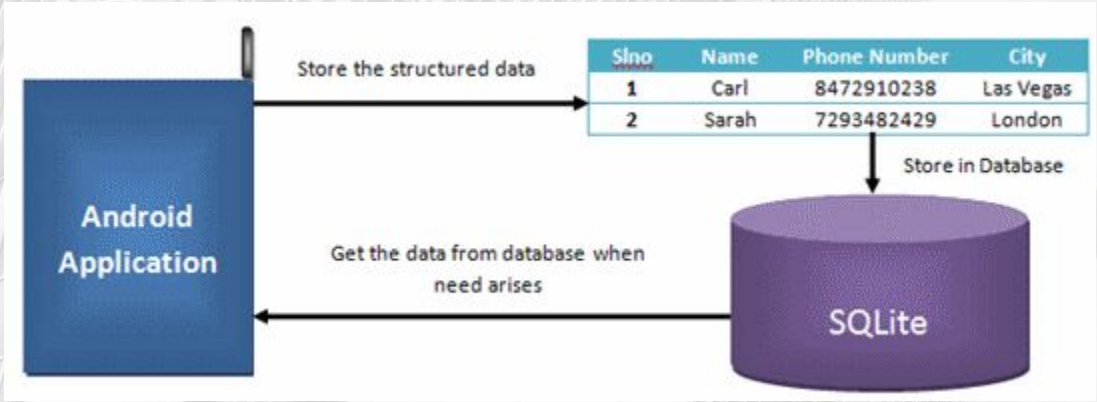
SQLite Database

SQLite merupakan sebuah *Relational Database Management System* (RDMS) yang tidak memerlukan server untuk beroperasi. Karena itu SQLite dapat digunakan dalam berbagai aplikasi *(offline)* seperti aplikasi Android, aplikasi desktop, dan game.

Komponen SQLite DB

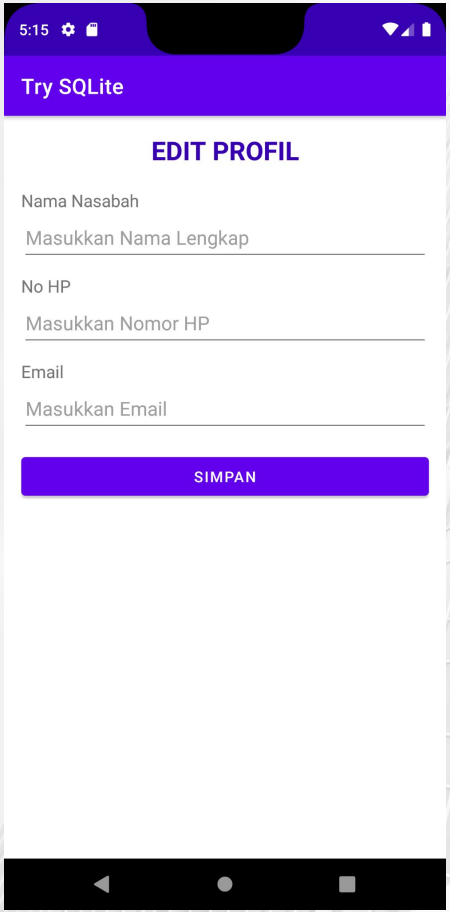
1. Class Model (Entity Data) : Representasi dari data yang akan disimpan ke dalam database

2. Database Handler Class : Class yang berfungsi untuk meng-handle operasi CRUD (Create, Read, Update, and Delete) dan meng-extends class **SQLiteOpenHelper**



Let’s try!

Membuat Aplikasi CRUD Sederhana



Aplikasi CRUD sederhana dengan SQLite yang menyimpan data umum nasabah pengguna mobile banking seperti nama lengkap nasabah, nomor HP, dan email.

Buatlah project baru di Android Studio terlebih dahulu yang terdiri dari satu empty activity!

1 : Membuat layout UI halaman edit profil (file activity_main.xml)

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:text="Edit Profil"
        android:textAlignment="center"
        android:textAllCaps="true"
        android:textColor="@color/purple_700"
        android:textSize="24sp"
        android:textStyle="bold" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nama Nasabah"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/et_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Masukkan Nama Lengkap"
        android:inputType="textPersonName"
        android:minHeight="48dp" />
```

```xml
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="No HP"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/et_nohp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Masukkan Nomor HP"
        android:inputType="number"
        android:minHeight="48dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="Email"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/et_address"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Masukkan Email"
        android:inputType="textEmailAddress"
        android:minHeight="48dp" />
```

```xml
    <Button
        android:id="@+id/btn_add"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dp"
        android:text="Simpan" />

</LinearLayout>
```

2 : Memberikan izin untuk mengakses penyimpanan pada file AndroidManifest.xml

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<application
```

3 : Membuat class model User

```
data class User(
    var id: Int = 0,
    var name: String,
    var phoneNum: String,
    var email : String
)
```

4 : Membuat class untuk operasi SQLite

```
class DatabaseHandler(context: Context?) :
    SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_VERSION = 1

        // Database name
        private const val DATABASE_NAME = "DB_APP"

        // table name
        private const val TABLE_USER = "USER"

        // column tables
        private const val KEY_ID = "id"
        private const val KEY_NAME = "name"
        private const val KEY_PHONE_NUM = "phone_num"
        private const val KEY_EMAIL = "email"
    }
```

Lanjutan :

```
    //Create table
    override fun onCreate(db: SQLiteDatabase) {
        val queryCreateTable = ("CREATE TABLE " + TABLE_USER + "("
                + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
                + KEY_PHONE_NUM + " TEXT" + KEY_EMAIL + " TEXT)")
        db.execSQL(queryCreateTable)
    }

    // on Upgrade database
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL( sql: "DROP TABLE IF EXISTS $TABLE_USER")
        onCreate(db)
    }
}
```

5 : Insert new data

```
fun insertUser(userModels: User) {
    val db = writableDatabase
    val values = ContentValues()
    values.put(KEY_NAME, userModels.name)
    values.put(KEY_PHONE_NUM, userModels.phoneNum)
    values.put(KEY_EMAIL, userModels.email)
    db.insert(TABLE_USER,  nullColumnHack: null, values)
    db.close()
}
```

6 : Read all data from DB

```kotlin
// get All Record
fun getAllUser(): List<User>? {
    val users: MutableList<User> = ArrayList<User>()
    // Select All Query
    val selectQuery = "SELECT  * FROM $TABLE_USER"
    val db = this.writableDatabase
    val cursor: Cursor = db.rawQuery(selectQuery,  selectionArgs: null)
    if (cursor.moveToFirst()) {
        do {
            val userModels = User(
                id = cursor.getString(0).toInt(),
                name = cursor.getString(1),
                phoneNum = cursor.getString(2),
                email = cursor.getString(3)
            )
            users.add(userModels)
        } while (cursor.moveToNext())
    }
    return users
}
```

7 : Delete data

```
fun deleteUser(user: User) {
    val db = this.writableDatabase
    db.delete(TABLE_USER, whereClause: "$KEY_ID = ?", arrayOf(String.valueOf(user.id)))
    db.close()
}
```

8 : Update data

```
fun updateUser(user: User): Int {
    val db = this.writableDatabase
    val values = ContentValues()
    values.put(KEY_NAME, user.name)
    values.put(KEY_PHONE_NUM, user.phoneNum)
    values.put(KEY_EMAIL, user.email)

    // updating data
    return db.update(
        TABLE_USER,
        values,
        whereClause: "$KEY_ID = ?",
        arrayOf(String.valueOf(user.id))
    )
}
```

9 : Akses DB dari activity

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.btnAdd.setOnClickListener {  it: View!
            val db = DatabaseHandler( context: this)
            val user = User(
                name = binding.etName.text.toString(),
                phoneNum = binding.etNohp.text.toString(),
                email = binding.etEmail.text.toString()
            )
            db.insertUser(user)
            Toast.makeText( context: this,  text: "Berhasil menambahkan ke dalam DB", Toast.LENGTH_LONG).show()
        }
    }
}
```

Berikut ini merupakan contoh pengaksesan DB Handler untuk insert data user baru.


Thank You!
mandiri