Flutter: Android Studioda ma'lumotlar bazasi bilan ishlash, SQLite, MySQL MBBT lar bilan ulanish

Reja:

1. Flutter: Android Studioda ma'lumotlar bazasi bilan ishlash
2. SQLite, MySQL MBBT lar bilan ulanish

SQLite ma'lumotlar bazasiga ulanish

Android ma'lumotlar bazasini boshqarishning eng keng tarqalgan tizimlaridan biri **SQLite**-ni qo'llab-quvvatlaydi. **android.database.sqlite** paketi **SQLite** ma'lumotlar bazalari bilan ishlashga imkon beradigan sinflar to'plamini aniqlaydi. Va har bir dastur o'z ma'lumot bazasini yaratishi mumkin.

Androidda **SQLite**-dan foydalanish uchun SQL tili yordamida ma'lumotlar bazasini yaratish kerak. Shundan so'ng ma'lumotlar bazasi yo'l bo'ylab dastur katalogida saqlanadi:

DATA/data/[Ilova_nomi]/databases/[ma'lumotlar_bazasi_fayli_nomi].

Standart ravishda, Android OS allaqachon standart dasturlar tomonidan ishlatiladigan bir nechta o'rnatilgan SQLite to’plamlarini o'z ichiga oladi - kontaktlar ro'yxati, kameradan fotosuratlarni saqlash, musiqiy albomlar va hk.

Ma'lumotlar bazalari bilan ishlashning asosiy funksiyalari **android.database** paketi tomonidan ta'minlanadi. To'g'ridan-to'g'ri **SQLite** bilan ishlash uchun **android.database.sqlite** paketi mavjud.

SQLitedagi ma'lumotlar bazasi **android.database.sqlite.SQLiteDatabase** klassi yordamida ifodalanadi. Bu ma'lumotlar bazasiga so'rovlarni bajarishga, u bilan turli xil manipulyatsiyalarni bajarishga imkon beradi.

android.database.sqlite.SQLiteCursor sinfi so'rovni taqdim etadi va shu so'rovga mos qatorlar to'plamini qaytarishga imkon beradi.

android.database.sqlite.SQLiteQueryBuilder klassi SQL so'rovlarini yaratishga imkon beradi.

android.database.sqlite.SQLiteOpenHelper klassi, agar ular mavjud bo'lmasa, barcha jadvallar bilan ma'lumotlar bazasini yaratishga imkon beradi.

SQLite quyidagi ma'lumotlar turi tizimidan foydalanadi:

- **INTEGER**: javadagi int turiga o'xshash butun sonni ifodalaydi.
- **REAL**: javada float va double -ga o'xshash raqamni ifodalaydi.
- **TEXT**: javada String va char -ga o'xshash belgilar majmuini ifodalaydi.
- **BLOB**: binar ma'lumotlarning massivini ifodalaydi.

Ma'lumotlar bazasini yaratish va ochish.

Androidda Activity kodida yangi ma'lumotlar bazasini yaratish yoki ochish uchun **openOrCreateDatabase()** metodini chaqirishimiz mumkin. Ushbu metod uchta parametrni olishi mumkin:

- ma'lumotlar bazasi nomi.
- ishlash rejimini aniqlaydigan raqamli qiymat (odatda **MODE_PRIVATE** doimiysi shaklida).
- ma'lumotlar bazasi bilan ishlash uchun kursor yaratish factorysi-ni belgilaydigan **SQLiteDatabase.CursorFactory** ob'ekti ko'rinishidagi ixtiyoriy parametr.

Masalan, **app.db** ma'lumotlar bazasini yaratish:

```
SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE_PRIVATE, null);
```

Ma'lumotlar bazasi so'rovini bajarish uchun **SQLiteDatabase** sinfining ***execSQL*** usulidan foydalanishingiz mumkin.

Masalan, ma'lumotlar bazasida **users** jadvalini yaratish:

```
SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE_PRIVATE, null);
db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age INTEGER)");
```

Ma'lumotlar bazasidan ba'zi ma'lumotlarni olish kerak bo'lsa, unda rawQuery() metodi qo'llaniladi.

Masalan, ma'lumotlar bazasidan barcha ob’ektlarni olish:

```
SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE_PRIVATE, null);
db.execSQL("CREATE TABLE IF NOT EXISTS users (ismi TEXT, yoshi INTEGER)");
Cursor query = db.rawQuery("SELECT * FROM users;", null);
if(query.moveToFirst()){
    String ismi = query.getString(0);
    int yoshi = query.getInt(1);
}
```

db.rawQuery() metodi qabul qilingan ma'lumotlarni olishimiz mumkin bo'lgan Cursor ob'ektini qaytaradi.

Ma'lumotlar bazasida ob'ektlar bo'lmasligi mumkin va buning uchun ***query.moveToFirst()*** metodi yordamida ma'lumotlar bazasidan olingan birinchi ob'ektga o'tadi. Agar bu metod false qiymatini qaytarsa, demak so'rov ma'lumotlar bazasidan hech qanday ma'lumot olmagan.

Endi ma'lumotlar bazasi bilan ishlash uchun oddiy ilova tuzamiz. Buning uchun Android Studioda yangi project yarating.

activity_main.xml faylini eng sodda grafik interfeys uchun quyidagicha o’zgartiring:

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

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

```
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="Click" />

    <TextView
        android:id="@+id/text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp" />

</LinearLayout>
```

MainActivity.java faylida biz ma'lumotlar bazasi bilan o'zaro aloqani bog’laymiz:

```
package com.example.android.sqlite;

import androidx.appcompat.app.AppCompatActivity;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

```
    public void onClick(View view) {
        SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE_PRIVATE, null);
        db.execSQL("CREATE TABLE IF NOT EXISTS users (ismi TEXT, yoshi INTEGER)");
        db.execSQL("INSERT INTO users VALUES ('Alisher', 24);");
        db.execSQL("INSERT INTO users VALUES ('Bahrom', 30);");

        Cursor query = db.rawQuery("SELECT * FROM users;", null);
        TextView textView = findViewById(R.id.text_view);
        if (query.moveToFirst()) {
            do {
                String ismi = query.getString(0);
                int yoshi = query.getInt(1);
                textView.append("Ismi: " + ismi + "    Yoshi: " + yoshi + "\n");
            }
            while (query.moveToNext());
        }
        query.close();
        db.close();
    }
}
```

Bu ilovadagi tugmani bosish bilan avval *app.db* ma'lumotlar bazasida ***users*** yangi jadvali yaratiladi, so'ngra unga **INSERT** yordamida ma'lumotlar bazasiga ikkita ob'ekt qo'shiladi.

Keyinchalik, **SELECT** yordamida barcha qo'shilgan foydalanuvchilarni ma'lumotlar bazasidan Cursor kursori ko'rinishida olamiz.

query.moveToFirst()-ni chaqirib, biz kursorni birinchi ob'ektga o'tkazamiz va bir nechta ob'ektga ega bo'lishimiz mumkinligi sababli, **do ... while** siklida hamma kursor bosib o'tiladi.

Kursordan ma'lumot olish uchun **query.getString(0)** va **query.getInt(1)** metodlaridan foydalaniladi. Ma'lumotlarni oladigan ustunning nomeri qavs ichida metodlarga o'tkaziladi. Masalan, yuqorida biz avval foydalanuvchi nomini String tipida, so'ngra yoshni raqam(int) sifatida qo'shdik. Bu shuni anglatadiki, nolinchi ustun **getString()** metodi yordamida olingan String qiymati bo'ladi, va keyingi - birinchi ustun **getInt()** metodi qo'llaniladigan raqamli (int) qiymatdir.

Kursor va ma'lumotlar bazasi bilan ishlashni tugatgandan so'ng, biz barcha tegishli ob'ektlarni yopamiz:

```
query.close();
db.close();
```

Agar Cursorni yopmasak, xotirada muammo yuzaga kelishi mumkin.

Misol:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical" android:layout_width="fill_parent"
android:layout_height="fill_parent">
<TextView android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="@string/hello"
android:textSize="18dp" android:paddingBottom="10dp"/>
<TextView android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="Ismingiz:"
android:textSize="15dp"/>
```

<EditText android:id="@+id/ism" android:layout_width="157dp"
android:layout_height="wrap_content" /?
<TextView android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="Familyangiz:" />
<EditText android:id="@+id/familya" android:layout_width="163dp"
android:layout_height="wrap_content" />
<Button android:id="@+id/add" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="Add" />
<Button android:id="@+id/deleteall" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="Delete All" />
<ListView android:id="@+id/contentlist" android:layout_width="fill_parent"
android:layout_height="fill_parent"/>
>

Quyidagi ko’rinishda layout hosil bo’ladi:

Yana bitta row.xml fayl yaratamiz, bu fayl bizga nima uchun kerak? Bu fayl bizga bazamizdagi malumotlarni ko’rsatib berish uchun kerak.

row.xml

<?xml version="1.0" encoding="utf-8"?>

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical" android:layout_width="fill_parent"
android:layout_height="wrap_content“>
<LinearLayout android:orientation="horizontal"
android:layout_width="fill_parent" android:layout_height="wrap_content“>
<TextView android:layout_width="wrap_content"
android:layout_height="fill_parent" android:padding="2dip"
android:text="#"/>
<TextView android:id="@+id/id" android:layout_width="wrap_content"
android:layout_height="fill_parent" android:padding="2dip"
android:paddingRight="10dip"/>

<TextView android:layout_width="wrap_content"
android:layout_height="fill_parent" android:padding="2dip"
android:paddingRight="10dip" android:text="-" />
<TextView android:id="@+id/text1" android:layout_width="fill_parent"
android:layout_height="fill_parent" android:padding="2dip"/>
</LinearLayout>
<TextView android:id="@+id/text2" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:padding="2dip"/>
</LinearLayout>
```

Kerakli layout larni yaratib olingach, asosiy kod yoziladi. Activity faylimizni ochib, main.xml faylida ko’rsatgan editTextlar ,Buttonlar va listView larni activity faylimizga bog’laymiz.

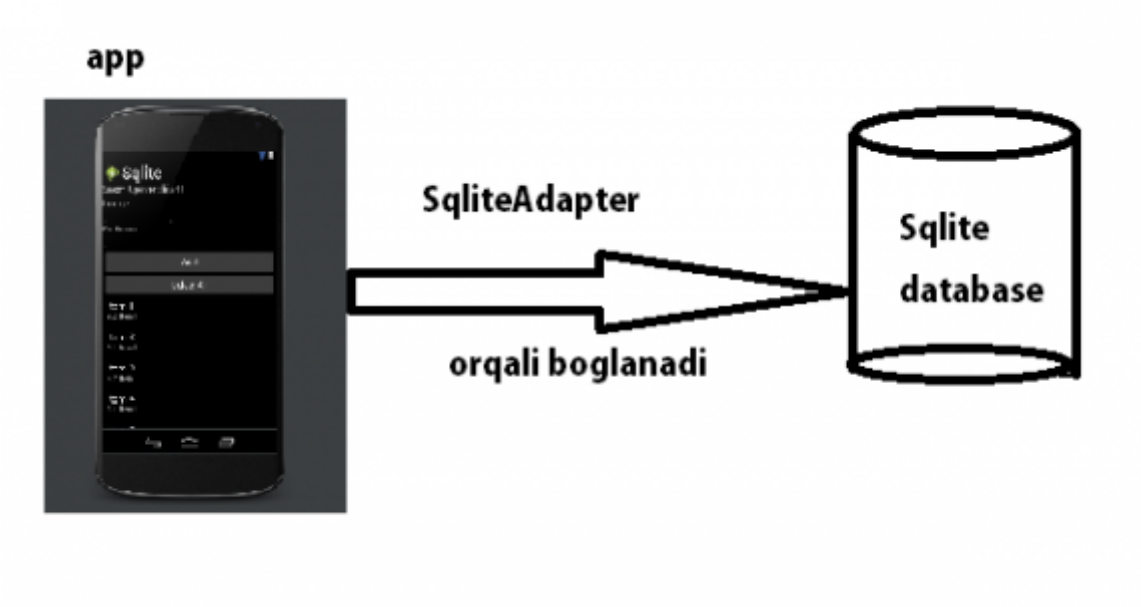
```
ism = (EditText)findViewById(R.id.ism);
familya= (EditText)findViewById(R.id.familya);
 buttonAdd = (Button)findViewById(R.id.add);
buttonDeleteAll = (Button)findViewById(R.id.deleteall);
```

opennetlist = (ListView)findViewById(R.id.opennetlist);

Endi biz dasturimizni malumotlar bazasi bilan bog’laymiz. Biz SqliteAdapter nomli klass yaratamiz, aynan shunday nom bo’lishi shart emas, o’ziz hohlagan nom berishiz mumkin.

Albatta Constructor ni yaratamiz.

```
 public SQLiteAdapter(Context c)
{ context = c;
   }
```



Ushbu klassimizni ichida yana bitta klass yaratamiz. Bu klassning nomini SQLiteHelper deb nomladik, va SQLiteOpenHelper klassidan extend olgan.

SQLiteOpenHelper — bu bizga database yaratib, undagi amallar bajarishga yordam beruvchi klass.

Bu klassning 2 ta metodi mavjud:

onCreate — database yaratish uchun

onUpgrade — yaratilgan database ni yangilash uchun

onCreate metodini ichiga quydagilarni yoziladi:

```
@Override public void onCreate(SQLiteDatabase db)
 { // TODO Auto-generated method stub
db.execSQL(SCRIPT_CREATE_DATABASE); }
```

Endi quyidagi o’zgaruvchilarni yaratamiz, bu o’zgaruvchilar database bilan ishlash uchun har doim kerak bo’ladigan o’zgaruvchilar bo’lganligi uchun hususiyatini public qilamiz

```
public static final String MYDATABASE_NAME = "opennet";
public static final String MYDATABASE_TABLE = "opennetchilar";
public static final int MYDATABASE_VERSION = 1;
public static final String KEY_ID = "_id";
 public static final String ism = "ism";
 public static final String familya = "familya";
//create table MY_DATABASE (ID integer primary key, Content text not null);
 private static final String SCRIPT_CREATE_DATABASE = "create table "
+ MYDATABASE_TABLE
 + " (" + KEY_ID + " integer primary key autoincrement, " + ism + " text not null, " + familya + " text not null);";
private SQLiteHelper sqLiteHelper;
    private SQLiteDatabase sqLiteDatabase;
    private Context context;
```

Endi database yaratish funksiyasiga kelsak, quyidagikodni yozamiz

```
 sqLiteHelper = new SQLiteHelper(context, MYDATABASE_NAME, null, MYDATABASE_VERSION);
```

SQLiteHelper klassiga murojaat qiladi va onCreate metodi ishga tushadi, va bizning databaseimizni yaratib beradi.

Eslatma:database ni 1 marta yaratadi,agar shunday nomli database bor bo’lsa yaratmaydi.

SqliteAdapter klassimizni ichida quyidagi funksiyalarni yaratamiz:

openToRead() — nomidan ham ma’lumki, databaseni o’qishga ruxsat olish uchun public SQLiteAdapter openToRead() throws android.database.SQLException { sqLiteHelper = new SQLiteHelper(context, MYDATABASE_NAME, null, MYDATABASE_VERSION); sqLiteDatabase = sqLiteHelper.getReadableDatabase(); return this; }

openToWrite() — nomidan ham ma’lumki, databaseni tahrirlash uchun ruxsat olish

```
public SQLiteAdapter openToWrite() throws android.database.SQLException {
sqLiteHelper = new SQLiteHelper(context, MYDATABASE_NAME, null, MYDATABASE_VERSION);
sqLiteDatabase = sqLiteHelper.getWritableDatabase();
 return this; }
```

Va nihoyat databaseni ishlatib bo’lganimizdan so’ng, uni yopish.

```
 public void close(){ sqLiteHelper.close();
}
```

Database ga ma’lumot qo’shish funksiyasi

```
 public long insert(String content1, String content2){ ContentValues contentValues = new ContentValues(); contentValues.put(ism, content1); contentValues.put(familya, content2);
return
sqLiteDatabase.insert(MYDATABASE_TABLE, null, contentValues); }
```

Database dagi barcha ma’lumotlarni o’chirish funksiyasi

```
public int deleteAll(){return sqLiteDatabase.delete(MYDATABASE_TABLE, null, null); }
```

Databasidagi barcha ma’lumotlarni olish uchun ishlatiladigan funksiya

```
public Cursor queueAll(){String[] columns = new String[]{KEY_ID, ism, familya};
Cursor cursor = sqLiteDatabase.query(MYDATABASE_TABLE, columns, null, null, null, null, null);
return cursor; }
```

Endi SqliteAdapter imiz dan yangi object yaratamiz va database ga yozish uchun ruxsat beruvchi funksiyani ishga tushiramiz:

```
 mySQLiteAdapter = new SQLiteAdapter(this); mySQLiteAdapter.openToWrite();
```

Database dagi barcha ma’lumotlarni oluvchi funksiyani ishga tushiramiz:

cursor = mySQLiteAdapter.queueAll(); SimpleCursorAdapter — bu Adapter bizga databasedagi malumotlarni ro'yhatda ko'rsatishga yordam beradi. Databasedan barcha ma'lumotlarni olganimizdan keyin uni ListView ga joylaymiz:

String[] from = new String[]{SQLiteAdapter.KEY_ID, SQLiteAdapter.ism, SQLiteAdapter.familya}; int[] to = new int[]{R.id.id, R.id.text1, R.id.text2}; cursorAdapter = new SimpleCursorAdapter(this, R.layout.row, cursor, from, to); opennetchilar.setAdapter(cursorAdapter);

Endi Button tugmalari bilan ishlash qoldi yani yangi ma'lumot qo'shish tugmasi va barcha ma'lumotlarni o'chirish tugmasi:

Button.OnClickListener buttonAddOnClickListener = new Button.OnClickListener(){ @Override public void onClick(View arg0)

{ // TODO Auto-generated method stub String data1 = ism.getText().toString();

String data2 = familya.getText().toString();

mySQLiteAdapter.insert(data1, data2);

updateList(); }

};

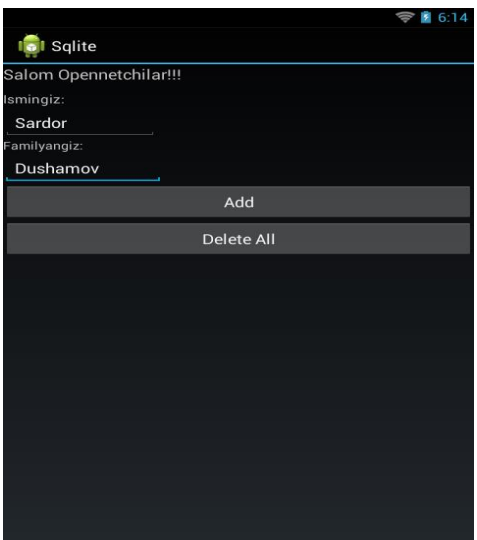
Button.OnClickListener buttonDeleteAllOnClickListener = new Button.OnClickListener(){ @Override public void onClick(View arg0)

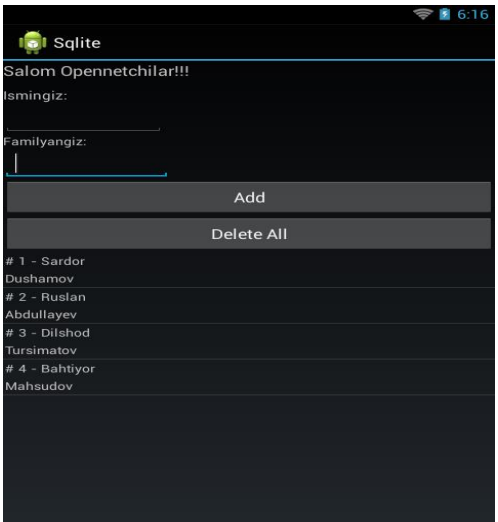
{ // TODO Auto-generated method stub mySQLiteAdapter.deleteAll();

updateList(); }

};

Endi Dasturni ishga tushiramiz va ma'lumot kiritib Add tugmasini bosamiz:





Nazorat savollari

1. Flutter: Android Studioda ma'lumotlar bazasi bilan ishlash haqida umumiy ma’lumot bering?
2. SQLite, MySQL MBBT lar bilan ulanish qayt taribda amalga oshiriladi?