В Android имеется встроенная поддержка одной из распространенных систем управления базами данных - SQLite. Для этого в пакете **android.database.sqlite** определен набор классов, которые позволяют работать с базами данных SQLite. И каждое приложение может создать свою базу данных.

Чтобы использовать SQLite в Android, надо создать базу данных с помощью выражение на языке SQL. После этого база данных будет храниться в каталоге приложения по пути:

```
DATA/data/[Название_приложения]/databases/[Название_файла_базы_данных]
```

ОС Android по умолчанию уже содержит ряд встроенных бад SQLite, которые используются стандартными программами - для списка контактов, для хранения фотографий с камеры, музыкальных альбомов и т.д.

Основную функциональность по работе с базами данных предоставляет пакет **android.database**. Функциональность непосредственно для работы с SQLite находится в пакете **android.database.sqlite**.

База данных в SQLite представлена классом **android.database.sqlite.SQLiteDatabase**. Он позволяет выполнять запросы к бд, выполнять с ней различные манипуляции.

Класс **android.database.sqlite.SQLiteCursor** предоставляет запрос и позволяет возвращать набор строк, которые соответствуют этому запросу.

Класс **android.database.sqlite.SQLiteQueryBuilder** позволяет создавать SQL-запросы.

Сами sql-выражения представлены классом **android.database.sqlite.SQLiteStatement**, которые позволяют с помощью плейсхолдеров вставлять в выражения динамические данные.

Класс **android.database.sqlite.SQLiteOpenHelper** позволяет создать базу данных со всеми таблицами, если их еще не существует.

В SQLite применяется следующая система типов данных:

- **INTEGER**: представляет целое число, аналог типу `int` в java

- **REAL**: представляет число с плавающей точкой, аналог `float` и `double` в java
- **TEXT**: представляет набор символов, аналог `String` и `char` в java
- **BLOB**: представляет массив бинарных данных, например, изображение, аналог типу `int` в java

Сохраняемые данные должны представлять соответствующие типы в java.

Создание и открытие базы данных

Для создания или открытия новой базы данных из кода Activity в Android мы можем вызвать метод **openOrCreateDatabase()**. Этот метод может принимать три параметра:

- название для базы данных
- числовое значение, которое определяет режим работы (как правило, в виде константы `MODE_PRIVATE`)
- необязательный параметр в виде объекта `SQLiteDatabase.CursorFactory`, который представляет фабрику создания курсора для работы с бд

Например, создание базы данных `app.db`:

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

Для выполнения запроса к базе данных можно использовать метод **execSQL** класса SQLiteDatabase. В этот метод передается SQL-выражение. Например, создание в базе данных таблицы users:

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

Если нам надо не просто выполнить выражение, но и получить из бд какие-либо данные, то используется метод **rawQuery()**. Этот метод в качестве параметра принимает SQL-выражение, а также набор значений для выражения sql. Например, получение всех объектов из базы данных:

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

```
Cursor query = db.rawQuery("SELECT * FROM users;", null);
if(query.moveToFirst()){

    String name = query.getString(0);
    int age = query.getInt(1);
}
```

Метод `db.rawQuery()` возвращает объект Cursor, с помощью которого мы можем извлечь полученные данные.

Возможна ситуация, когда в базе данных не будет объектов, и для этого методом `query.moveToFirst()` пытаемся переместиться к первому объекту, полученному из бд. Если этот метод возвратит значение false, значит запрос не получил никаких данных из бд.

Теперь для работы с базой данных сделаем простейшее приложение. Для этого создадим новый проект.

В файле **activity_main.xml** определим простейший графический интерфейс:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp" >

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click"
        android:onClick="onClick"
        app:layout_constraintBottom_toTopOf="@id/textView"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="22sp"
        app:layout_constraintTop_toBottomOf="@id/button"
        app:layout_constraintLeft_toLeftOf="parent"/>
```

```
26
27</androidx.constraintlayout.widget.ConstraintLayout>
```

А в классе **MainActivity** определим взаимодействие с базой данных:

```
1package com.example.sqliteapp;
2
3import androidx.appcompat.app.AppCompatActivity;
4
5import android.database.Cursor;
6import android.database.sqlite.SQLiteDatabase;
7import android.os.Bundle;
8import android.view.View;
9import android.widget.TextView;
10
11public class MainActivity extends AppCompatActivity {
12
13    @Override
14    protected void onCreate(Bundle savedInstanceState) {
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.activity_main);
17    }
18    public void onClick(View view){
19        SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db",
20MODE_PRIVATE, null);
21        db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age INTEGER,
22UNIQUE(name))");
23        db.execSQL("INSERT OR IGNORE INTO users VALUES ('Tom Smith', 23), ('John Dow',
2431);");
25
26        Cursor query = db.rawQuery("SELECT * FROM users;", null);
27        TextView textView = findViewById(R.id.textView);
28        textView.setText("");
29        while(query.moveToNext()){
30            String name = query.getString(0);
31            int age = query.getInt(1);
32            textView.append("Name: " + name + " Age: " + age + "\n");
33        }
34        query.close();
        db.close();
    }
}
```

По нажатию на кнопку здесь вначале создается в базе данных app.db новая таблица users, а затем в нее добавляются два объекта в базу данных с помощью SQL-выражения INSERT.

Далее с помощью выражения SELECT получаем всех добавленных пользователей из базы данных в виде курсора Cursor.

Вызовом `query.moveToNext()` перемещаемся в цикле `while` последовательно по всем объектам.

Для получения данных из курсора применяются методы `query.getString(0)` и `query.getInt(1)`. В скобках в методы передается номер столбца, из которого мы получаем данные. Например, выше мы добавили вначале имя пользователя в виде строки, а затем возраст в виде числа. Значит, нулевым столбцом будет идти строкое значение, которое получаем с помощью метода `getString()`, а следующим - первым столбцом идет числовое значение, для которого применяется метод `getInt()`.

После завершения работы с курсором и базой данных мы закрываем все связанные объекты:

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

Если мы не закроем курсор, то можем столкнуться с проблемой утечки памяти.

Для упрощения работы с базами данных SQLite в Android нередко применяется класс **SQLiteOpenHelper**. Для использования необходимо создать класса-наследник от SQLiteOpenHelper. Поэтому добавим в проект, в ту же папку, где находится класс MainActivity, новый класс **DatabaseHelper**

```
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase;
import android.content.Context;

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "userstore.db"; // название бд
    private static final int SCHEMA = 1; // версия базы данных
    static final String TABLE = "users"; // название таблицы в бд
    // названия столбцов
    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_NAME = "name";
    public static final String COLUMN_YEAR = "year";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, SCHEMA);
    }
```

```
    @Override
    public void onCreate(SQLiteDatabase db) {

        db.execSQL("CREATE TABLE users (" + COLUMN_ID
                + " INTEGER PRIMARY KEY AUTOINCREMENT," + COLUMN_NAME
                + " TEXT, " + COLUMN_YEAR + " INTEGER);");
        // добавление начальных данных
        db.execSQL("INSERT INTO "+ TABLE +" (" + COLUMN_NAME
                + ", " + COLUMN_YEAR  + ") VALUES ('Том Смит', 1981);");
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,  int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS "+TABLE);
        onCreate(db);
    }
}
```

Если база данных отсутствует или ее версия (которая задается в переменной SCHEMA) выше текущей, то срабатывает метод `onCreate()`.

Для выполнения запросов к базе данных нам потребуется объект **SQLiteDatabase**, который представляет базу данных. Метод `onCreate()` получает в качестве параметра базу данных приложения.

Для выполнения запросов к SQLite используется метод **execSQL()**. Он принимает sql-выражение CREATE TABLE, которое создает таблицу. Здесь также при необходимости мы можем выполнить и другие запросы, например, добавить какие-либо начальные данные. Так, в данном случае с помощью того же метода и выражения sql INSERT добавляется один объект в таблицу.

В методе `onUpgrade()` происходит обновление схемы БД. В данном случае для примера использован примитивный поход с удалением предыдущей базы данных с помощью sql-выражения DROP и последующим ее созданием. Но в реальности если вам будет необходимо сохранить данные, этот метод может включать более сложную логику - добавления новых столбцов, удаление ненужных, добавление дополнительных данных и т.д.

Создадим следующий класс, соотвествующий вашему entity:

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
```

```
import android.database.sqlite.SQLiteDatabase;
import java.util.ArrayList;
import java.util.List;

public class DatabaseAdapter {
    private DatabaseHelper dbHelper;
    private SQLiteDatabase database;

    public DatabaseAdapter(Context context) {
        dbHelper = new DatabaseHelper(context.getApplicationContext());
    }

    public DatabaseAdapter open() {
        database = dbHelper.getWritableDatabase();
        return this;
    }

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

    public List<User> getUsers() {
        List<User> users = new ArrayList<>();
        Cursor cursor = database.query(
                DatabaseHelper.TABLE,
                new String[]{DatabaseHelper.COLUMN_ID,
DatabaseHelper.COLUMN_NAME, DatabaseHelper.COLUMN_YEAR},
                null, null, null, null, null
        );

        while (cursor.moveToNext()) {
            long id =
cursor.getLong(cursor.getColumnIndexOrThrow(DatabaseHelper.COLUMN_ID));
            String name =
cursor.getString(cursor.getColumnIndexOrThrow(DatabaseHelper.COLUMN_NAME));
            int year =
cursor.getInt(cursor.getColumnIndexOrThrow(DatabaseHelper.COLUMN_YEAR));
            users.add(new User(id, name, year));
        }
        cursor.close();
        return users;
    }

    public long insert(User user) {
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, user.getName());
```

```
        cv.put(DatabaseHelper.COLUMN_YEAR, user.getYear());
        return database.insert(DatabaseHelper.TABLE, null, cv);
    }

    public long update(User user) {
        ContentValues cv = new ContentValues();
        cv.put(DatabaseHelper.COLUMN_NAME, user.getName());
        cv.put(DatabaseHelper.COLUMN_YEAR, user.getYear());
        return database.update(
                DatabaseHelper.TABLE,
                cv,
                DatabaseHelper.COLUMN_ID + " = ?",
                new String[]{String.valueOf(user.getId())}
        );
    }

    public long delete(long id) {
        return database.delete(
                DatabaseHelper.TABLE,
                DatabaseHelper.COLUMN_ID + " = ?",
                new String[]{String.valueOf(id)}
        );
    }

    public User getUser(long id) {
        Cursor cursor = database.query(
                DatabaseHelper.TABLE,
                null,
                DatabaseHelper.COLUMN_ID + " = ?",
                new String[]{String.valueOf(id)},
                null, null, null
        );

        User user = null;
        if (cursor.moveToFirst()) {
            String name =
cursor.getString(cursor.getColumnIndexOrThrow(DatabaseHelper.COLUMN_NAME));
            int year =
cursor.getInt(cursor.getColumnIndexOrThrow(DatabaseHelper.COLUMN_YEAR));
            user = new User(id, name, year);
        }
        cursor.close();
        return user;
    }
}
```

Этот класс представляет собой простейший CRUD, позволяющий вам достать данные из БД.

Добавьте в ваш Fragment данный код

```
private void loadUsers() {
    dbAdapter = new DatabaseAdapter(this).open();
    List<User> users = dbAdapter.getUsers();
    dbAdapter.close();
    adapter.updateUsers(users);
}

@Override
protected void onResume() {
    super.onResume();
    loadUsers();
}
```

Это позволит вам подгружать список из бд.

Задание.

Переработайте вашу activity с фрагментами. Оставьте один FragmentContainer, в котором будет выводиться RecyclerView с вашим entity. Добавьте кнопку для добавления вашей сущности в таблицу. Сделайте переход на другой фрагмент по нажатию на элемент с помощью:

```
NextFragment nextFrag= new NextFragment();
Bundle args = new Bundle();
nextFrag.setArguments(args);
getActivity().getSupportFragmentManager().beginTransaction()
            .replace(R.id.Layout_container, nextFrag, "findThisFragment")
            .addToBackStack(null)
            .commit();
```

Сделайте, чтобы туда выводилась информация об вашей Entity, которую можно будет менять в таблице, а также удалять.

Сделайте, чтобы у вас выводился список entity, сохранённых в таблице.