

**КАЛУЖСКИЙ ФИЛИАЛ  
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО  
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ИМЕНИ Н.Э. БАУМАНА (национальный исследовательский университет)»**



**Факультет** «Информатика и управление»

**Кафедра** "Программное обеспечение ЭВМ, информационные технологии"

# **Хранение данных в Android-приложениях. Базы данных**

**Калуга**

# Хранение данных. SQLite

В приложении, при подключении к БД, мы указываем имя БД и версию. При этом могут возникнуть следующие ситуации:

**1.БД не существует.** Это может быть например в случае первичной установки программы. В этом случае приложение должно само создать БД и все таблицы в ней. И далее оно уже работает с только что созданной БД.

**2.БД существует, но ее версия устарела.** Это может быть в случае обновления программы. В этом случае приложение должно обновить существующие таблицы и создать новые, если это необходимо.

**3.БД существует и ее версия актуальна.** В этом случае приложение успешно подключается к БД и работает.



Для обработки описанных выше ситуаций необходимо создать класс, являющийся наследником для **SQLiteOpenHelper**.

В лекционных примерах он будет называться **DBHelper**. Этот класс предоставляет нам методы для создания или обновления БД в случаях ее отсутствия или устаревания.

- **onCreate** - метод, который будет вызван, если БД, к которой мы хотим подключиться – не существует

- **onUpgrade** - будет вызван в случае, если мы пытаемся подключиться к БД более новой версии, чем существующая

**Разработаем простое приложение, которое будет хранить имя и возраст.** Ввод данных будет осуществляться на экране приложения, а для отображения информации будут использоваться логи.

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

```
<LinearLayout
    android:id="@+id/linearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:text="Name" >
</TextView>
```

```
<EditText
    android:id="@+id/etName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1" >
</EditText>
```

```
</LinearLayout>
```

```
<LinearLayout
    android:id="@+id/linearLayout3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
```

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:text="Age" >
```

```
</TextView>
```

```
<EditText
    android:id="@+id/etAge"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1" >
```

```
</EditText>
</LinearLayout>
```

```
<LinearLayout
    android:id="@+id/linearLayout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
```

```
<Button
    android:id="@+id/btnAdd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Add" >
</Button>
```

```
<Button
    android:id="@+id/btnRead"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Read" >
</Button>
```

```
<Button
    android:id="@+id/btnClear"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Clear" >
```

```
</Button>
</LinearLayout>
```

```
</LinearLayout>
```

## //Java

```
import android.app.Activity;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity implements OnClickListener {
    final String LOG_TAG = "myLogs";
    Button btnAdd, btnRead, btnClear;
    EditText etName, etAge;
    DBHelper dbHelper;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnAdd = (Button) findViewById(R.id.btnAdd);
        btnAdd.setOnClickListener(this);
        btnRead = (Button) findViewById(R.id.btnRead);
        btnRead.setOnClickListener(this);
        btnClear = (Button) findViewById(R.id.btnClear);
        btnClear.setOnClickListener(this);
        etName = (EditText) findViewById(R.id.etName);
        etAge = (EditText) findViewById(R.id.etAge);
        // создаем объект для создания и управления версиями БД
        dbHelper = new DBHelper(this);
    }
```

```
public void onClick(View v) {
// создаем объект для данных
ContentValues cv = new ContentValues();
// получаем данные из полей ввода
String name = etName.getText().toString(); String age = etAge.getText().toString();
// подключаемся к БД
SQLiteDatabase db = dbHelper.getWritableDatabase();
switch (v.getId()) {
case R.id.btnAdd:
Log.d(LOG_TAG, "--- Insert in mytable: ---");
// подготовим данные для вставки в виде пар: наименование столбца - значение
cv.put("name", name); cv.put("age", age);
// вставляем запись и получаем ее ID
long rowID = db.insert("mytable", null, cv);
Log.d(LOG_TAG, "row inserted, ID = " + rowID);
break;
case R.id.btnRead:
Log.d(LOG_TAG, "--- Rows in mytable: ---");
// делаем запрос всех данных из таблицы mytable, получаем Cursor
Cursor c = db.query("mytable", null, null, null, null, null, null);
// ставим позицию курсора на первую строку выборки если в выборке нет строк, вернется false
if (c.moveToFirst()) { // определяем номера столбцов по имени в выборке
int idColIndex = c.getColumnIndex("id");
int nameColIndex = c.getColumnIndex("name");
int ageColIndex = c.getColumnIndex("age");
do { // получаем значения по номерам столбцов и пишем все в лог
Log.d(LOG_TAG,
"ID = " + c.getInt(idColIndex) + ", name = "
+ c.getString(nameColIndex) + ", age = "
+ c.getString(ageColIndex));
// переход на следующую строку а если следующей нет (текущая - последняя), то false -
// выходим из цикла
} while (c.moveToNext());
} else
Log.d(LOG_TAG, "0 rows");
break;
```

```

case R.id.btnClear:
Log.d(LOG_TAG, "--- Clear mytable: ---");
// удаляем все записи
int clearCount = db.delete("mytable", null, null);
Log.d(LOG_TAG, "deleted rows count = " + clearCount);
break;
}
// закрываем подключение к БД
dbHelper.close();
}

class DBHelper extends SQLiteOpenHelper {
public DBHelper(Context context) {
// конструктор суперкласса
super(context, "myDB", null, 1);
}

@Override
public void onCreate(SQLiteDatabase db) {
Log.d(LOG_TAG, "--- onCreate database ---");
// создаем таблицу с полями
db.execSQL("create table mytable ("
+ "id integer primary key autoincrement," + "name text,"
+ "age text" + ");");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
}
}
}

```

# //Kotlin

```
import android.content.ContentValues
import android.content.Context
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.EditText
import kotlinx.android.synthetic.main.activity_main.*
import android.database.sqlite.SQLiteDatabase
import android.util.Log

class MainActivity : AppCompatActivity() {

    val SQL_CREATE_TABLE : String = "create table if not exists mytable (id integer primary key
autoincrement," +
        " name text, age text);"
    val LOG_TAG = "myLogs";

    private var db: SQLiteDatabase? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        btnAdd.setOnClickListener(this::onClick)
        btnRead.setOnClickListener(this::onClick)
        btnClear.setOnClickListener(this::onClick)
        //Подключаемся к БД
        db = openOrCreateDatabase("app.db", Context.MODE_PRIVATE, null)
        db!!.execSQL(SQL_CREATE_TABLE)
    }

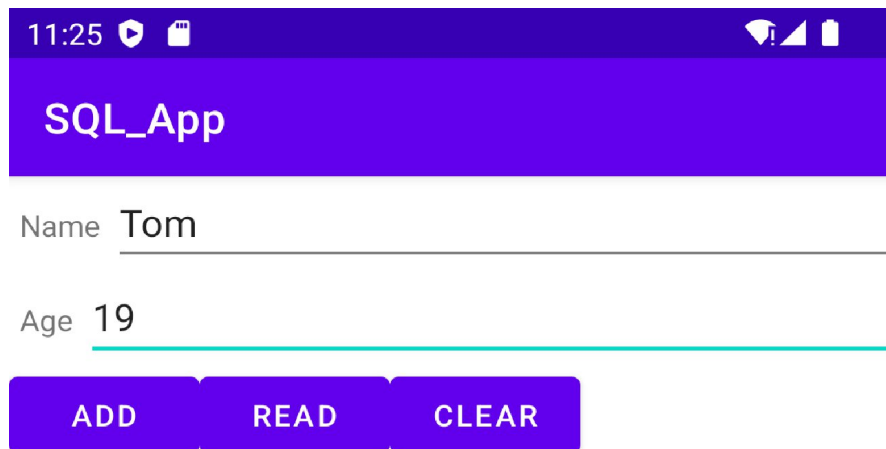
    fun onClick(view: View){
        // создаем объект для данных
        var cv = ContentValues()
        // получаем данные из полей ввода
        var etName = findViewById(R.id.etName) as EditText
        var etAge = findViewById(R.id.etAge) as EditText
        var name = etName.getText().toString()
        var age = etAge.getText().toString()
```



```
when (view.id){
    R.id.btnAdd -> {
        Log.d(LOG_TAG, "--- Insert in mytable: ---")
        // подготовим данные для вставки в виде пар: наименование столбца - значение
        cv.put("name", name)
        cv.put("age", age)
        // вставляем запись и получаем ее ID
        var rowID = db!!.insert("mytable", null, cv);
        Log.d(LOG_TAG, "row inserted, ID = " + rowID)
    }
    R.id.btnRead -> {
        Log.d(LOG_TAG, "--- Rows in mytable: ---");
        // делаем запрос всех данных из таблицы mytable, получаем Cursor
        var c = db!!.query("mytable", null, null, null,
            null, null, null)
        if (c.moveToFirst()) {
            var idColIndex = c.getColumnIndex("id")
            var nameColIndex = c.getColumnIndex("name")
            var ageColIndex = c.getColumnIndex("age")
            do { // получаем значения по номерам столбцов и пишем все в лог
                Log.d(LOG_TAG, "ID = " + c.getInt(idColIndex) + ", name = "
                    + c.getString(nameColIndex) + ", age = "
                    + c.getString(ageColIndex))
            } while (c.moveToNext());
        } else
            Log.d(LOG_TAG, "0 rows")

    }
    R.id.btnClear -> {
        Log.d(LOG_TAG, "--- Clear mytable: ---");
        // удаляем все записи
        var clearCount = db!!.delete("mytable", null, null)
        Log.d(LOG_TAG, "deleted rows count = " + clearCount)
    }
}
}
```

```
10-17 05:39:45.135: D/myLogs(25275): --- onCreate database ---
10-17 05:39:45.145: D/myLogs(25275): --- Insert in mytable: ---
10-17 05:39:45.155: D/myLogs(25275): row inserted, ID = 1
10-17 05:40:40.905: D/myLogs(25275): --- Insert in mytable: ---
10-17 05:40:40.915: D/myLogs(25275): row inserted, ID = 2
10-17 05:40:53.835: D/myLogs(25275): --- Rows in mytable: ---
10-17 05:40:53.835: D/myLogs(25275): ID = 1, name = John, age = 23
10-17 05:40:53.835: D/myLogs(25275): ID = 2, name = Kate, age = 25
10-17 05:41:26.275: D/myLogs(25275): --- Insert in mytable: ---
10-17 05:41:26.285: D/myLogs(25275): row inserted, ID = 3
10-17 05:41:28.535: D/myLogs(25275): --- Rows in mytable: ---
10-17 05:41:28.545: D/myLogs(25275): ID = 1, name = John, age = 23
10-17 05:41:28.545: D/myLogs(25275): ID = 2, name = Kate, age = 25
10-17 05:41:28.545: D/myLogs(25275): ID = 3, name = Tom, age = 19
10-17 05:41:51.395: D/myLogs(25275): --- Clear mytable: ---
10-17 05:41:51.405: D/myLogs(25275): deleted rows count = 3
```



11:25

SQL\_App

Name Tom

Age 19

ADD READ CLEAR

- В методе **Activity - onCreate** определяются объекты, присваиваются обработчики и создается объект dbHelper класса DBHelper для управления БД.
- В методе **Activity - onClick**, в котором обрабатываются нажатия на кнопки.

Класс **ContentValues** используется для указания полей таблицы и значений, которые в эти поля будут вноситься.

Далее записываются в переменные значения из полей ввода. Затем, с помощью метода **getWritableDatabase** происходит подключение к БД и получение объекта SQLiteDatabase.

В приложении будут использоваться его методы: insert – вставка записи, query – чтение, delete – удаление.

**//Java**

```
public void onClick(View v) {
    // создаем объект для данных
    ContentValues cv = new ContentValues();
    // получаем данные из полей ввода
    String name = etName.getText().toString(); String age = etAge.getText().toString();
    // подключаемся к БД
    SQLiteDatabase db = dbHelper.getWritableDatabase();
```

**//Kotlin**

```
fun onClick(view: View){
    // создаем объект для данных
    var cv = ContentValues()
    // получаем данные из полей ввода
    var etName = findViewById(R.id.etName) as EditText
    var etAge = findViewById(R.id.etAge) as EditText
    var name = etName.getText().toString()
    var age = etAge.getText().toString()
    //Подключаемся к БД
    db = openOrCreateDatabase("app.db", Context.MODE_PRIVATE, null)
    db!!.execSQL(SQL_CREATE_TABLE)
```

**btnAdd** – добавление записи в таблицу *mytable*.

1.Объект *cv* заполняется парами: имя поля и значение. Поле *id* заполнится автоматически (primary key autoincrement).

2.Вызывается метод *insert* – ему передается ему имя таблицы и объект *cv* с вставляемыми значениями. Второй аргумент метода используется, при вставке в таблицу пустой строки. В приложении это не нужно, поэтому передается *null*.

3.Метод *insert* возвращает ID вставленной строки, он сохраняется в *rowID* и выводится в лог.

### //Java

```
switch (v.getId()) {  
    case R.id.btnAdd:  
        Log.d(LOG_TAG, "--- Insert in mytable: ---");  
        // подготовим данные для вставки в виде пар: наименование столбца - значение  
        cv.put("name", name); cv.put("age", age);  
        // вставляем запись и получаем ее ID  
        long rowID = db.insert("mytable", null, cv);  
        Log.d(LOG_TAG, "row inserted, ID = " + rowID);  
        break;
```

### //Kotlin

```
when (view.id) {  
    R.id.btnAdd -> {  
        Log.d(LOG_TAG, "--- Insert in mytable: ---")  
        // подготовим данные для вставки в виде пар: наименование столбца - значение  
        cv.put("name", name)  
        cv.put("age", age)  
        // вставляем запись и получаем ее ID  
        var rowID = db!!.insert("mytable", null, cv);  
        Log.d(LOG_TAG, "row inserted, ID = " + rowID)  
    }  
}
```

**btnRead** – чтение всех записей из таблицы *mytable*.

1. Для чтения используется метод `query`. На вход ему подается имя таблицы, список запрашиваемых полей, условия выборки, группировка, сортировка. Т.к. требуются все данные во всех полях без сортировок и группировок, то используется везде `null`.
2. Метод возвращает объект класса `Cursor`. Его можно рассматривать как таблицу с данными.
3. Метод `moveToFirst` – делает первую запись в `Cursor` активной и проверяет, есть ли вообще записи в нем (т.е. выбралось ли что-либо в методе `query`).
4. Далее возвращаются порядковые номера столбцов в `Cursor` по их именам с помощью метода `getColumnIndex`. Эти номера затем используются для чтения данных в методах `getInt` и `getString` и вывода данных в лог.
5. С помощью метода `moveToNext` перебираются все строки в `Cursor` пока не достигается последняя. Если же записей не было, то в лог выводится соответствующее сообщение – *0 rows*.

## //Java

```
case R.id.btnRead:
Log.d(LOG_TAG, "--- Rows in mytable: ---");
Cursor c = db.query("mytable", null, null, null,
null, null, null);
if (c.moveToFirst()) { // определяем номера столбцов
по имени в выборке
int idColIndex = c.getColumnIndex("id");
int nameColIndex = c.getColumnIndex("name");
int emailColIndex = c.getColumnIndex("age");
do { // получаем значения по номерам столбцов и пишем
все в лог
Log.d(LOG_TAG,
"ID = " + c.getInt(idColIndex) + ", name = "
+ c.getString(nameColIndex) + ", age = "
+ c.getString(emailColIndex));
// переход на следующую строку а если следующей нет
(текущая - последняя), то false - выходим из цикла
} while (c.moveToNext());
} else Log.d(LOG_TAG, "0 rows"); break;
```

## //Kotlin

```
R.id.btnRead -> {
    Log.d(LOG_TAG, "--- Rows in mytable: ---");
    var c = db!!.query("mytable", null, null, null,
        null, null, null)
    if (c.moveToFirst()) {
        var idColIndex = c.getColumnIndex("id")
        var nameColIndex = c.getColumnIndex("name")
        var ageColIndex = c.getColumnIndex("age")
        do { // получаем значения по номерам столбцов
            и пишем все в лог
            Log.d(LOG_TAG, "ID = " +
                c.getInt(idColIndex) + ", name = "
                + c.getString(nameColIndex) + ", age = "
                + c.getString(ageColIndex))
        } while (c.moveToNext());
    } else
        Log.d(LOG_TAG, "0 rows")
}
```

**btnClear** – очистка таблицы.

- 1.Метод delete удаляет записи. На вход передаем имя таблицы и null в качестве условий для удаления.
- 2.Метод возвращает количество удаленных записей.
- 3.После этого закрывается соединение с БД методом close.

Класс DBHelper является вложенным в MainActivity. Этот класс должен наследовать класс SQLiteOpenHelper.

**//Java**

```
case R.id.btnClear:
Log.d(LOG_TAG, "--- Clear mytable: ---");
// удаляем все записи
int clearCount = db.delete("mytable", null, null);
Log.d(LOG_TAG, "deleted rows count = " + clearCount);
break;
}
// закрываем подключение к БД
dbHelper.close();
}
```

**//Kotlin**

```
R.id.btnClear -> {
    Log.d(LOG_TAG, "--- Clear mytable: ---");
    // удаляем все записи
    var clearCount = db!!.delete("mytable", null, null)
    Log.d(LOG_TAG, "deleted rows count = " + clearCount)
}
```

В конструкторе вызывается конструктор суперкласса и ему передаются:

- context - контекст
- mydb* - название базы данных
- null – объект для работы с курсорами, нам пока не нужен, поэтому null
- 1 – версия базы данных

В методе onCreate этого класса используется метод execSQL объекта SQLiteDatabase для выполнения SQL-запроса, который создает таблицу. Этот метод вызывается, если БД не существует и ее надо создавать. По запросу видно, что создается таблица *mytable* с полями *id*, *name* и *age*.

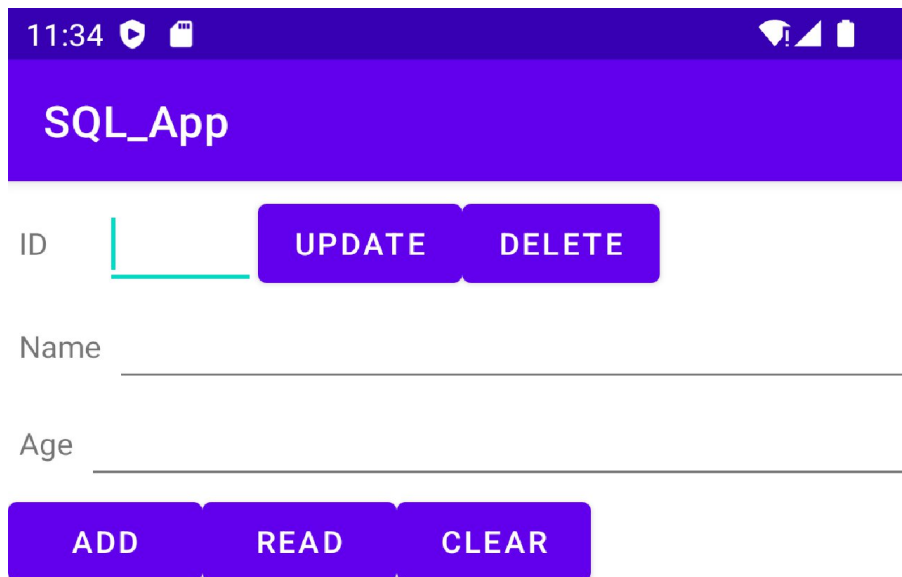
Метод onUpgrade пока не заполняется т.к. используется одна версия БД.

```
class DBHelper extends SQLiteOpenHelper {
public DBHelper(Context context) {
// конструктор суперкласса
super(context, "myDB", null, 1);
}

@Override
public void onCreate(SQLiteDatabase db) {
Log.d(LOG_TAG, "--- onCreate database ---");
// создаем таблицу с полями
db.execSQL("create table mytable ("
+ "id integer primary key autoincrement," + "name text,"
+ "age text" + ");");
}
```

# Обновление и удаление записей

Перейдем к **обновлению и удалению конкретной записи**. На основе предыдущего проекта немного поменяем экран, добавим поле для ввода ID и кнопки для обновления и удаления. По нажатию кнопки Update считывается содержимое полей Name и Age, и происходит обновление записи в таблице, для которой id равно значению из поля ID. По нажатию кнопки Delete удаляется запись из таблицы по id равному значению из поля ID.



The screenshot shows a mobile application interface titled "SQL\_App". At the top, there is a status bar with the time 11:34 and various icons. Below the title bar, there is a section for updating and deleting records. It includes an "ID" label next to a text input field. To the right of the input field are two buttons: "UPDATE" and "DELETE". Below the ID section, there are two more text input fields labeled "Name" and "Age". At the bottom of the interface, there are three buttons: "ADD", "READ", and "CLEAR".



```

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

    <LinearLayout
        android:id="@+id/linearLayout4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp" >

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dp"
            android:layout_marginRight="25dp"
            android:text="ID" >
        </TextView>

        <EditText
            android:id="@+id/etID"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="2dp" >
        </EditText>

        <Button
            android:id="@+id/btnUpd"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Update" >
        </Button>

        <Button
            android:id="@+id/btnDel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Delete" >
        </Button>
    </LinearLayout>

```

```

<LinearLayout
    android:id="@+id/linearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:text="Name" >
    </TextView>

    <EditText
        android:id="@+id/etName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1" >
    </EditText>

</LinearLayout>

```

```
<LinearLayout
    android:id="@+id/linearLayout3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:text="Age" >
    </TextView>

    <EditText
        android:id="@+id/etAge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1" >
    </EditText>
</LinearLayout>
```

```
<LinearLayout
    android:id="@+id/linearLayout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <Button
        android:id="@+id/btnAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add" >
    </Button>

    <Button
        android:id="@+id/btnRead"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Read" >
    </Button>

    <Button
        android:id="@+id/btnClear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clear" >
    </Button>
</LinearLayout>

</LinearLayout>
```

## //Java

```
public class MainActivity extends Activity implements OnClickListener {
    final String LOG_TAG = "myLogs";
    Button btnAdd, btnRead, btnClear, btnUpd, btnDel;
    EditText etName, etAge, etID;
    DBHelper dbHelper;

    /** Called when the activity is first created. */
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnAdd = (Button) findViewById(R.id.btnAdd);
        btnAdd.setOnClickListener(this);
        btnRead = (Button) findViewById(R.id.btnRead);
        btnRead.setOnClickListener(this);
        btnClear = (Button) findViewById(R.id.btnClear);
        btnClear.setOnClickListener(this);
        btnUpd = (Button) findViewById(R.id.btnUpd);
        btnUpd.setOnClickListener(this);
        btnDel = (Button) findViewById(R.id.btnDel);
        btnDel.setOnClickListener(this);
        etName = (EditText) findViewById(R.id.etName);
        etAge = (EditText) findViewById(R.id.etAge);
        etID = (EditText) findViewById(R.id.etID);
        // создаем объект для создания и управления версиями БД
        dbHelper = new DBHelper(this);
    }

    public void onClick(View v) {
        // создаем объект для данных
        ContentValues cv = new ContentValues();
        // получаем данные из полей ввода
        String name = etName.getText().toString();
        String age = etAge.getText().toString();
        String id = etID.getText().toString();
        // подключаемся к БД
        SQLiteDatabase db = dbHelper.getWritableDatabase();
    }
}
```

```

switch (v.getId()) {
case R.id.btnAdd:
Log.d(LOG_TAG, "--- Insert in mytable: ---");
// подготовим данные для вставки в виде пар: наименование столбца -
// значение
cv.put("name", name);
cv.put("Age", age);
// вставляем запись и получаем ее ID
long rowID = db.insert("mytable", null, cv);
Log.d(LOG_TAG, "row inserted, ID = " + rowID);
break;

case R.id.btnRead:
Log.d(LOG_TAG, "--- Rows in mytable: ---");
// делаем запрос всех данных из таблицы mytable, получаем Cursor
Cursor c = db.query("mytable", null, null, null, null, null, null);
// ставим позицию курсора на первую строку выборки
// если в выборке нет строк, вернется false
if (c.moveToFirst()) {
// определяем номера столбцов по имени в выборке
int idColIndex = c.getColumnIndex("id");
int nameColIndex = c.getColumnIndex("name");
int ageColIndex = c.getColumnIndex("Age");
do {
// получаем значения по номерам столбцов и пишем все в лог
Log.d(LOG_TAG,
"ID = " + c.getInt(idColIndex) + ", name = "
+ c.getString(nameColIndex) + ", Age = "
+ c.getString(ageColIndex));
// переход на следующую строку
// а если следующей нет (текущая - последняя), то false -
// выходим из цикла
} while (c.moveToNext());
} else
Log.d(LOG_TAG, "0 rows");
break;

```

```

case R.id.btnClear:
Log.d(LOG_TAG, "---- Clear mytable: ---");
// удаляем все записи
int clearCount = db.delete("mytable", null, null);
Log.d(LOG_TAG, "deleted rows count = " + clearCount);
break;

case R.id.btnUpd:
if (id.equalsIgnoreCase("")) {
break;
}
Log.d(LOG_TAG, "---- Update mytabe: ---");
// подготовим значения для обновления
cv.put("name", name);
cv.put("Age", age);
// обновляем по id
int updCount = db.update("mytable", cv, "id = ?",
new String[] { id });
Log.d(LOG_TAG, "updated rows count = " + updCount);
break;

case R.id.btnDel:
if (id.equalsIgnoreCase("")) {
break;
}
Log.d(LOG_TAG, "---- Delete from mytabe: ---");
// удаляем по id
int delCount = db.delete("mytable", "id = " + id, null);
Log.d(LOG_TAG, "deleted rows count = " + delCount);
break;
}
// закрываем подключение к БД
dbHelper.close();
}

```

```

class DBHelper extends SQLiteOpenHelper {
public DBHelper(Context context) {
// конструктор суперкласса
super(context, "myDB", null, 1);
}

public void onCreate(SQLiteDatabase db) {
Log.d(LOG_TAG, "---- onCreate database ---");
// создаем таблицу с полями
db.execSQL("create table mytable ("
+ "id integer primary key autoincrement," + "name text,"
+ "age text" + ");");
}

public void onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion) {
}
}

```

## //Kotlin

```
class MainActivity : AppCompatActivity() {

    val SQL_CREATE_TABLE : String = "create table IF NOT EXISTS mytable (id integer primary
key autoincrement," +
        " name text, age text);"
    val LOG_TAG = "myLogs";
    private var db: SQLiteDatabase? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        btnAdd.setOnClickListener(this::onClick)
        btnRead.setOnClickListener(this::onClick)
        btnClear.setOnClickListener(this::onClick)
        btnUpd.setOnClickListener(this::onClick)
        btnDel.setOnClickListener(this::onClick)
        //Подключаемся к БД
        db = openOrCreateDatabase("app.db", Context.MODE_PRIVATE, null)
        db!!.execSQL(SQL_CREATE_TABLE)
    }

    fun onClick(view: View){
        // создаем объект для данных
        var cv = ContentValues()
        // получаем данные из полей ввода
        var etName = findViewById(R.id.etName) as EditText
        var etAge = findViewById(R.id.etAge) as EditText
        var name = etName.getText().toString()
        var age = etAge.getText().toString()
        var id = etID.getText().toString()
    }
}
```

```

when (view.id){
    R.id.btnAdd -> {
        Log.d(LOG_TAG, "--- Insert in mytable: ---")
        // подготовим данные для вставки в виде пар: наименование столбца - значение
        cv.put("name", name)
        cv.put("age", age)
        // вставляем запись и получаем ее ID
        var rowID = db!!.insert("mytable", null, cv);
        Log.d(LOG_TAG, "row inserted, ID = " + rowID)
    }
    R.id.btnRead -> {
        Log.d(LOG_TAG, "--- Rows in mytable: ---");
        // делаем запрос всех данных из таблицы mytable, получаем Cursor
        var c = db!!.query("mytable", null, null, null,
            null, null, null)
        if (c.moveToFirst()) {
            var idColIndex = c.getColumnIndex("id")
            var nameColIndex = c.getColumnIndex("name")
            var ageColIndex = c.getColumnIndex("age")
            do { // получаем значения по номерам столбцов и пишем все в лог
                Log.d(LOG_TAG, "ID = " + c.getInt(idColIndex) + ", name = "
                    + c.getString(nameColIndex) + ", age = "
                    + c.getString(ageColIndex))
                // переход на следующую строку а если следующей нет (текущая - последняя), то
                false -
                // выходим из цикла
            } while (c.moveToNext());
        } else
            Log.d(LOG_TAG, "0 rows")
    }
}

```

```

R.id.btnClear -> {
    Log.d(LOG_TAG, "--- Clear mytable: ---");
    // удаляем все записи
    var clearCount = db!!.delete("mytable", null, null)
    Log.d(LOG_TAG, "deleted rows count = " + clearCount)
}
R.id.btnUpd -> {
    if ((id as java.lang.String).equalsIgnoreCase("")) {
        return
    }
    Log.d(LOG_TAG, "--- Update mytable: ---")
    // подготовим значения для обновления
    cv.put("name", name)
    cv.put("Age", age);
    // обновляем по id
    var updCount = db!!.update("mytable", cv, "id = ?", arrayOf(id))
    Log.d(LOG_TAG, "updated rows count = " + updCount)
}
R.id.btnDel -> {
    if ((id as java.lang.String).equalsIgnoreCase("")) {
        return
    }
    Log.d(LOG_TAG, "--- Delete from mytabe: ---")
    // удаляем по id
    var delCount = db!!.delete("mytable", "id = " + id, null)
    Log.d(LOG_TAG, "deleted rows count = " + delCount)
}
}
}
}

```



В код добавлена переменная `id`, в нее заносится значение поля `etID`. В `switch` добавляются две новые метки:

•**btnUpd** – обновление записи в `mytable`. Проверка того, что значение `id` не пустое, заполнение `cv` данными для обновления и обновление записи. Для этого используется метод `update`. На вход ему подается имя таблицы, заполненный `ContetnValues` с значениями для обновления, строка условия (`Where`) и массив аргументов для строки условия. В строке условия использовался знак `'?'`. При запросе к БД вместо этого знака будет подставлено значение из массива аргументов. Метод `update` возвращает количество обновленных записей, которое выводится в лог.

```
String id = etID.getText().toString();  
int updCount = db.update("mytable", cv, "id = ?", new String[] { id });
```

•**btnDel** – удаление записи из `mytable`. Проверка того, что `id` не пустое и вызов метода `delete`. На вход передается имя таблицы, строка условия и массив аргументов для условия. Метод `delete` возвращает количество удаленных строк, которое выводится в лог.

```
int delCount = db.delete("mytable", "id = " + id, null);
```

Обратите внимание, что условия и для `update` и для `delete` одинаковые, а именно `id =`

## btnUpd

### //Java

```
case R.id.btnUpd:
if (id.equalsIgnoreCase("")) {
break;
}
Log.d(LOG_TAG, "--- Update mytabe: ---");
// подготовим значения для обновления
cv.put("name", name);
cv.put("Age", age);
// обновляем по id
int updCount = db.update("mytable", cv,
"id = ?",
new String[] { id });
Log.d(LOG_TAG, "updated rows count = " +
updCount);
break;
```

### //Kotlin

```
R.id.btnUpd -> {
    if ((id as
java.lang.String).equalsIgnoreCase("")) {
        return
    }
    Log.d(LOG_TAG, "--- Update mytable: --
-")
    // подготовим значения для обновления
    cv.put("name", name)
    cv.put("Age", age);
    // обновляем по id
    var updCount = db!!.update("mytable",
cv, "id = ?", arrayOf(id))
    Log.d(LOG_TAG, "updated rows count = "
+ updCount)
}
```

## btnDel

### //Java

```
case R.id.btnDel:
if (id.equalsIgnoreCase("")) {
break;
}
Log.d(LOG_TAG, "--- Delete from mytabe: ---");
// удаляем по id
int delCount = db.delete("mytable", "id = " +
id, null);
Log.d(LOG_TAG, "deleted rows count = " +
delCount);
break;
}
// закрываем подключение к БД
dbHelper.close();
}
```

### //Kotlin

```
R.id.btnDel -> {
    if ((id as
java.lang.String).equalsIgnoreCase("")) {
        return
    }
    Log.d(LOG_TAG, "--- Delete from mytabe: -
-")
    // удаляем по id
    var delCount = db!!.delete("mytable", "id
= " + id, null)
    Log.d(LOG_TAG, "deleted rows count = " +
delCount)
}
```

# SQLite. Подробнее про метод query.

## Условие, сортировка, группировка

В прошлом проекте использовался метод query для чтения всех данных из таблицы. Использовалось только имя таблицы в качестве входного параметра и извлекались все записи.

**У метода query есть и другие параметры:**

- columns – список полей, которые необходимо получить
- selection – строка условия WHERE
- selectionArgs – массив аргументов для selection. В selection можно использовать знаки ?, которые будут заменены этими значениями.
- groupBy - группировка
- having – использование условий для агрегатных функций
- orderBy – сортировка

**Рассмотрим приложение – справочник вузов.** Возьмем 15 вузов и сохраним в БД их наименование, количество студентов и страну. Реализуем следующие функции:

оывовод всех записей

оывовод значения агрегатной функции (SUM, MIN, MAX, COUNT)

оывовод вузов с числом студентов, больше чем указано

огруппировка вузов по стране

оывовод стран с числом студентов больше, чем указано

осортировка вузов по наименованию, численности студентов и стране

Все данные выводятся в лог.

## layout-файл activity\_main.xml

Для сортировки используется RadioGroup.

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="5dp"
        android:layout_marginTop="5dp"
        android:gravity="center_horizontal"
        android:text="Университеты мира"
        android:textSize="14sp" >
    </TextView>

    <Button
        android:id="@+id/btnAll"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:text="Все записи" >
    </Button>

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp" >

        <Button
            android:id="@+id/btnFunc"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Функция" >
        </Button>

        <EditText
            android:id="@+id/etFunc"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1" >

            <requestFocus>
            </requestFocus>
        </EditText>
    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp" >

        <Button
            android:id="@+id/btnStudent"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Число студентов" >
        </Button>

        <EditText
            android:id="@+id/etStudent"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="number" >
        </EditText>
    </LinearLayout>
```

```

<Button
    android:id="@+id/btnGroup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:text="Студентов в стране" >
</Button>

<LinearLayout
    android:id="@+id/linearLayout4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp" >

    <Button
        android:id="@+id/btnHaving"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Студентов в стране" > >
    </Button>

    <EditText
        android:id="@+id/etCountryStudent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="number" >
    </EditText>
</LinearLayout>

<LinearLayout
    android:id="@+id/linearLayout3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp" >

```

```

<Button
    android:id="@+id/btnSort"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Сортировка" >
</Button>

<RadioGroup
    android:id="@+id/rgSort"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <RadioButton
        android:id="@+id/rName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="Название ВУЗа" >
    </RadioButton>

    <RadioButton
        android:id="@+id/rStudent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Число студентов" >
    </RadioButton>

    <RadioButton
        android:id="@+id/rCountry"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Страна" >
    </RadioButton>
</RadioGroup>
</LinearLayout>

</LinearLayout>

```

## //Java

```
public class MainActivity extends Activity implements
OnClickListener {
    final String LOG_TAG = "myLogs";
    String name[] = { "Кембридж", "Тюбеген", "Оксфорд",
        "Гейдельбург", "Мюнхен", "Гарвард", "МТИ", "Сорбона",
        "Принстон", "Имперский колледж", "Страсбург", "Беркли",
        "МГУ", "МГУ", "ТГУ" };
    int student[] = { 140, 211, 195, 142, 128, 82, 280, 60,
        66, 35, 223, 56, 110, 45, 121 };
    String country[] = { "Англия", "Германия", "Англия",
        "Герсания", "Германия", "США", "США", "Франция", "США",
        "Англия", "Франция", "США", "Россия", "Россия",
        "Россия" };
    Button btnAll, btnFunc, btnStudent, btnSort, btnGroup,
    btnHaving;
    EditText etFunc, etStudent, etCountryStudent;
    RadioGroup rgSort;
    DBHelper dbHelper;
    SQLiteDatabase db;
```

```
/** Called when the activity is first created. */
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    btnAll = (Button) findViewById(R.id.btnAll);
    btnAll.setOnClickListener(this);
    btnFunc = (Button) findViewById(R.id.btnFunc);
    btnFunc.setOnClickListener(this);
    btnStudent = (Button) findViewById(R.id.btnStudent);
    btnStudent.setOnClickListener(this);
    btnSort = (Button) findViewById(R.id.btnSort);
    btnSort.setOnClickListener(this);
    btnGroup = (Button) findViewById(R.id.btnGroup);
    btnGroup.setOnClickListener(this);
    btnHaving = (Button) findViewById(R.id.btnHaving);
    btnHaving.setOnClickListener(this);
    etFunc = (EditText) findViewById(R.id.etFunc);
    etStudent = (EditText) findViewById(R.id.etStudent);
    etCountryStudent = (EditText)
    findViewById(R.id.etCountryStudent);
    rgSort = (RadioGroup) findViewById(R.id.rgSort);
    dbHelper = new DBHelper(this);
```

```
// подключаемся к базе
db = dbHelper.getWritableDatabase();
// проверка существования записей
Cursor c = db.query("studTable", null, null, null,
    null, null, null);
if (c.getCount() == 0) {
    ContentValues cv = new ContentValues();
    // заполним таблицу
    for (int i = 0; i < 15; i++) {
        cv.put("name", name[i]);
        cv.put("student", student[i]);
        cv.put("country", country[i]);
        Log.d(LOG_TAG, "id = " + db.insert("studTable",
            null, cv));
    }
}
dbHelper.close();
// эмулируем нажатие кнопки btnAll
onClick(btnAll);
}
```

```

public void onClick(View v) {
    // подключаемся к базе
    db = dbHelper.getWritableDatabase();
    // данные с экрана
    String sFunc = etFunc.getText().toString();
    String sStudent = etStudent.getText().toString();
    String sCountryStudent =
etCountryStudent.getText().toString();
    // переменные для query
    String[] columns = null;
    String selection = null;
    String[] selectionArgs = null;
    String groupBy = null;
    String having = null;
    String orderBy = null;
    // курсор
    Cursor c = null;
    // определяем нажатую кнопку
    switch (v.getId()) {
        // Все записи
        case R.id.btnAll:
            Log.d(LOG_TAG, "--- Все записи ---");
            c = db.query("studTable", null, null, null, null, null,
null);
            break;
        // функция
        case R.id.btnFunc:
            Log.d(LOG_TAG, "--- функция " + sFunc + " ---");
            columns = new String[] { sFunc };
            c = db.query("studTable", columns, null, null, null,
null, null);
            break;
        // Студентов больше, чем
        case R.id.btnStudent:
            Log.d(LOG_TAG, "--- Студентов больше " + sStudent + " ---
");
            selection = "student > ?";
            selectionArgs = new String[] { sStudent };
            c = db.query("studTable", null, selection, selectionArgs,
null,
null, null);
            break;

```

```

// Студентов в стране
        case R.id.btnGroup:
            Log.d(LOG_TAG, "--- Студентов в стране ---");
            columns = new String[] { "country", "sum(student) as
student" };
            groupBy = "country";
            c = db.query("studTable", columns, null, null,
groupBy, null, null);
            break;
        // Студентов в стране больше чем
        case R.id.btnHaving:
            Log.d(LOG_TAG, "--- Страны с числом студентов больше
" + sCountryStudent
+ " ---");
            columns = new String[] { "country", "sum(student) as
student" };
            groupBy = "country";
            having = "sum(student) > " + sCountryStudent;
            c = db.query("studTable", columns, null, null,
groupBy, having,
null);
            break;
        // Сортировка
        case R.id.btnSort:
            // сортировка по
            switch (rgSort.getCheckedRadioButtonId()) {
                // название вуза
                case (R.id.rName):
                    Log.d(LOG_TAG, "--- Сортировка по названию вуза ---
");
                    orderBy = "name";
                    break;
                // число студентов
                case (R.id.rStudent):
                    Log.d(LOG_TAG, "--- Сортировка по студентам ---");
                    orderBy = "student";
                    break;
                // страна
                case (R.id.rCountry):
                    Log.d(LOG_TAG, "--- Сортировка по стране ---");
                    orderBy = "country";
                    break;
            }

```

```



c = db.query("studTable", null, null,
null, null, null, orderBy);
break;
}
if (c != null) {
if (c.moveToFirst()) {
String str;
do {
str = "";
for (String cn : c.getColumnNames()) {
str = str.concat(cn + " = "
+ c.getString(c.getColumnIndex(cn)) + "; ");
}
Log.d(LOG_TAG, str);
} while (c.moveToNext());
}
} else
Log.d(LOG_TAG, "Cursor is null");
dbHelper.close();
}

class DBHelper extends SQLiteOpenHelper {
public DBHelper(Context context) {
// конструктор суперкласса
super(context, "myDB", null, 1);
}

public void onCreate(SQLiteDatabase db) {
Log.d(LOG_TAG, "--- onCreate database ---");
// создаем таблицу с полями
db.execSQL("create table studTable ("
+ "id integer primary key autoincrement," + "name text,"
+ "student integer," + "country text" + ");");
}

public void onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion) {
}}}

```

7:48



## SQL\_App

Университеты мира

ВСЕ ЗАПИСИ

ФУНКЦИЯ

ЧИСЛО СТУДЕНТОВ >

СТУДЕНТОВ В СТРАНЕ

СТУДЕНТОВ В СТРАНЕ >

СОРТИРОВКА

☒ Название ВУЗа
☐ Число студентов
☐ Страна



# //Kotlin

```
class MainActivity : AppCompatActivity() {

    val LOG_TAG = "myLogs"
    val name = arrayOf("Кембридж", "Тюбеген", "Оксфорд", "Гейдельбург", "Мюнхен", "Гарвард", "МТИ",
"Сорбона", "Принстон", "Имперский колледж", "Страсбург", "Беркли", "МГТУ", "МГУ", "ТГУ")
    val student = arrayOf(140, 211, 195, 142, 128, 82, 280, 60, 66, 35, 223, 56, 110, 45, 121)
    val country = arrayOf("Англия", "Германия", "Англия", "Герсания", "Германия", "США", "США",
"Франция", "США", "Англия", "Франция", "США", "Россия", "Россия", "Россия")

    val SQL_CREATE_TABLE : String = "create table IF NOT EXISTS studTable (id integer primary key +
        " autoincrement, name text, student integer, country text);"
    private var db: SQLiteDatabase? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        btnAll.setOnClickListener(this::onClick)
        btnFunc.setOnClickListener(this::onClick)
        btnStudent.setOnClickListener(this::onClick)
        btnSort.setOnClickListener(this::onClick)
        btnGroup.setOnClickListener(this::onClick)
        btnHaving.setOnClickListener(this::onClick)
        db = openOrCreateDatabase("myDB", Context.MODE_PRIVATE, null)
        db!!.execSQL(SQL_CREATE_TABLE)
        var c = db!!.query("studTable", null, null, null,
            null, null, null)
        if (c.count == 0) {
            val cv = ContentValues()
            // заполним таблицу
            for (i in 0..14) {
                cv.put("name", name[i])
                cv.put("student", student[i])
                cv.put("country", country[i])
                Log.d(LOG_TAG, "id = " + db!!.insert("studTable", null, cv))
            }
        }
        onClick(btnAll)
    }
}
```

```

fun onClick(view: View){
    // данные с экрана
    var etFunk = findViewById(R.id.etFunc) as EditText
    var etStudent = findViewById(R.id.etStudent) as EditText
    var etCountryStudent = findViewById(R.id.etCountryStudent) as EditText
    var sFunc = etFunk.text.toString()
    var sStudent = etStudent.text.toString()
    var sCountryStudent = etCountryStudent.text.toString()
    // переменные для query
    var columns: Array<String>? = null
    var selection: String? = null
    var selectionArgs: Array<String>? = null
    var groupBy: String? = null
    var having: String? = null
    var orderBy: String? = null
    // курсор
    var c: Cursor? = null
    // определяем нажатую кнопку
    when(view.id){
        // Все записи
        R.id.btnAll -> {
            Log.d(LOG_TAG, "--- Все записи ---")
            c = db!!.query("studTable", null, null, null,
                null, null, null)
        }
        // функция
        R.id.btnFunc -> {
            Log.d(LOG_TAG, "--- функция " + sFunc + " ---")
            columns = arrayOf(sFunc)
            c = db!!.query("studTable", columns, null, null,
                null, null, null)
        }
    }
}

```

```
// Студентов больше, чем
```

```
R.id.btnStudent -> {  
    Log.d(LOG_TAG, "--- Студентов больше " + sStudent + " ---")  
    selection = "student > ?"  
    selectionArgs = arrayOf(sStudent)  
    c = db!!.query("studTable", null, selection, selectionArgs, null,  
        null, null)  
}
```

```
// Студентов в стране
```

```
R.id.btnGroup -> {  
    Log.d(LOG_TAG, "--- Студентов в стране ---")  
    columns = arrayOf("country", "sum(student) as student")  
    groupBy = "country"  
    c = db!!.query("studTable", columns, null, null, groupBy,  
        null, null)  
}
```

```
// Студентов в стране больше чем
```

```
R.id.btnHaving -> {  
    Log.d(LOG_TAG, "--- Страны с числом студентов больше " + sCountryStudent + " ---")  
    columns = arrayOf("country", "sum(student) as student")  
    groupBy = "country"  
    having = "sum(student) > $sCountryStudent"  
    c = db!!.query("studTable", columns, null, null,  
        groupBy, having, null  
    )  
}
```

// Сортировка

```
R.id.btnSort -> {  
    var rgSort = findViewById(R.id.rgSort) as RadioGroup  
    when (rgSort.getCheckedRadioButtonId()) {  
        // название вуза  
        R.id.rName -> {  
            Log.d(LOG_TAG, "--- Сортировка по названию вуза ---")  
            orderBy = "name"  
        }  
        // число студентов  
        R.id.rStudent -> {  
            Log.d(LOG_TAG, "--- Сортировка по студентам ---")  
            orderBy = "student"  
        }  
        // страна  
        R.id.rCountry -> {  
            Log.d(LOG_TAG, "--- Сортировка по стране ---")  
            orderBy = "country"  
        }  
    }  
    c = db!!.query("studTable", null, null, null,  
        null, null, orderBy)  
}  
  
if (c != null) {  
    if (c.moveToFirst()) {  
        var str = ""  
        do {  
            str = ""  
            for (cn in c.columnNames) {  
                str = str + cn + " = " + c.getString(c.getColumnIndex(cn)) + "; "  
            }  
            Log.d(LOG_TAG, str)  
        } while (c.moveToNext())  
    }  
}  
  
else  
    Log.d(LOG_TAG, "Cursor is null")  
}
```

Три массива данных name, student, country. Это наименования вузов, численность студентов (в тысячах) и страны, к которым относятся вузы. По этим данным заполняется таблица.

- В методе onCreate определяются и находятся экранные элементы, присваиваются обработчики, создается объект dbHelper для управления БД, подключение к базе данных и получение объекта db для работы с БД, проверка наличия записей в таблице, если нет ничего – заполняется данными, закрывается соединение и эмулируется нажатие кнопки «Все записи» для того, чтобы сразу вывести в лог весь список.

- В методе onClick производится подключение к базе, чтение данных с экранных полей в переменные, описание переменных, которые используются в методе query, и курсор. Далее определяется какая кнопка была нажата.

- btnAll – вывод всех записей. Вызов метода query с именем таблицы и null для остальных параметров.

**//Java**

```
case R.id.btnAll:
    Log.d(LOG_TAG, "--- Все записи ---");
    c = db.query("studTable", null, null, null, null, null, null);
    break;
```

**//Kotlin**

```
R.id.btnAll -> {
    Log.d(LOG_TAG, "--- Все записи ---")
    c = db!!.query("studTable", null, null, null,
        null, null, null)
}
```

**btnFunc** – вывод значения агрегатной функции (или любого поля). Используется параметр **columns**, в который надо записать поля, которые необходимо получить из таблицы, т.е. то, что обычно перечисляется после слова **SELECT** в SQL-запросе. **columns** имеет тип **String[]** – массив строк. Создание массива из одного значения, которое считано с поля **etFunc** на экране.

### //Java

```
case R.id.btnFunc:
Log.d(LOG_TAG, "--- функция " +
sFunc + " ---");
columns = new String[] { sFunc };
c = db.query("studTable", columns, null,
null, null, null, null);
break;
```

### //Kotlin

```
R.id.btnFunc -> {
    Log.d(LOG_TAG, "--- функция " +
sFunc + " ---")
    columns = arrayOf(sFunc)
    c = db!!.query("studTable",
columns, null, null,
null, null, null)
}
```

**btnStudent** – вывод вузов с числом студентов больше введенного на экране количества. Используется **selection** для формирования условия. При этом используем один аргумент - **?**. Значение аргумента задается в **selectionArgs** – это **sStudent** – содержимое поля **etStudent**.

### //Java

```
case R.id.btnStudent:
Log.d(LOG_TAG, "--- Студентов больше " +
sStudent + " ---");
selection = "student > ?";
selectionArgs = new String[] { sStudent };
c = db.query("studTable", null, selection,
selectionArgs, null,null, null);
break;
```

### //Kotlin

```
R.id.btnStudent -> {
    Log.d(LOG_TAG, "--- Студентов больше " +
sStudent + " ---")
    selection = "student > ?"
    selectionArgs = arrayOf(sStudent)
    c = db!!.query("studTable", null, selection,
selectionArgs, null, null, null)
}
```

**btnGroup** – группировка вузов по странам и вывод общего количества студентов. Используется `columns` для указания столбцов, которые необходимо получить – страна и общее число студентов. В `groupBy` указывается, что группировка будет по стране.

### //Java

```
case R.id.btnGroup:
    Log.d(LOG_TAG, "---- Студентов в стране ----");
    columns = new String[] { "country",
        "sum(student) as student" };
    groupBy = "country";
    c = db.query("studTable", columns, null, null,
        groupBy, null, null);
    break;
```

### //Kotlin

```
R.id.btnGroup -> {
    Log.d(LOG_TAG, "---- Студентов в стране ----")
    columns = arrayOf("country", "sum(student) as
student")
    groupBy = "country"
    c = db!!.query("studTable", columns, null,
null, groupBy,
        null, null)
}
```

**btnHaving** – вывод стран с числом студентов больше указанного числа. Полностью аналогично случаю с группировкой, но добавляется условие в параметре `having` – общее число студентов в стране должна быть меньше `sCountryStudent` (значение `etCountryStudent` с экрана).

### //Java

```
case R.id.btnHaving:
    Log.d(LOG_TAG, "---- Страны с числом студентов
больше " + sCountryStudent
+ " ----");
    columns = new String[] { "country", "sum(student)
as student" };
    groupBy = "country";
    having = "sum(student) > " + sCountryStudent;
    c = db.query("studTable", columns, null, null,
        groupBy, having,
        null);
    break;
```

### //Kotlin

```
R.id.btnHaving -> {
    Log.d(LOG_TAG, "---- Страны с числом
студентов больше " + sCountryStudent + " ----")
    columns = arrayOf("country", "sum(student)
as student")
    groupBy = "country"
    having = "sum(student) > $sCountryStudent"
    c = db!!.query("studTable", columns, null,
null,
        groupBy, having, null)
}
```

**btnSort** – сортировка стран. Определяем какой RadioButton включен и соответственно указываем в orderBy поле для сортировки данных.

В выше описанных случаях запускался query и получался объект с класса Cursor. Далее осуществлялась проверка, что он существует и в нем есть записи (moveToFirst). Если так, то запускается перебор записей в цикле do ... while (с.moveToNext()). Для каждой записи перебираются названия полей (getColumnNames), получаем по каждому полю его номер и извлекаем данные методом getString. Формируется список полей и значений в переменную str, которая потом выводится в лог. После всего этого закрывается соединение.

## //Java

```
case R.id.btnSort:
// сортировка по
switch (rgSort.getCheckedRadioButtonId()) {
// название вуза
case (R.id.rName):
Log.d(LOG_TAG, "--- Сортировка по названию вуза ---");
orderBy = "name";
break;
// число студентов
case (R.id.rStudent):
Log.d(LOG_TAG, "--- Сортировка по студентам ---");
orderBy = "student";
break;
// страна
case (R.id.rCountry):
Log.d(LOG_TAG, "--- Сортировка по стране ---");
orderBy = "country";
break;
}
c = db.query("studTable", null, null, null, null,
null, orderBy);
break;
}
```

## //Kotlin

```
R.id.btnSort -> {
    var rgSort = findViewById(R.id.rgSort) as
    RadioGroup
    when (rgSort.getCheckedRadioButtonId()) {
        // название вуза
        R.id.rName -> {
            Log.d(LOG_TAG, "--- Сортировка по
            названию вуза ---")
            orderBy = "name"
        }
        // число студентов
        R.id.rStudent -> {
            Log.d(LOG_TAG, "--- Сортировка по
            студентам ---")
            orderBy = "student"
        }
        // страна
        R.id.rCountry -> {
            Log.d(LOG_TAG, "--- Сортировка по стране
            ---")
            orderBy = "country"
        }
    }
    c = db!!.query("studTable", null, null, null,
    null, null, orderBy)
}
```



11-22 14:24:32.973: D/myLogs(414): --- onCreate database ---  
11-22 14:24:33.242: D/myLogs(414): id = 1  
11-22 14:24:33.332: D/myLogs(414): id = 2  
11-22 14:24:33.392: D/myLogs(414): id = 3  
11-22 14:24:33.411: D/myLogs(414): id = 4  
11-22 14:24:33.502: D/myLogs(414): id = 5  
11-22 14:24:33.593: D/myLogs(414): id = 6  
11-22 14:24:34.822: D/myLogs(414): id = 7  
11-22 14:24:34.911: D/myLogs(414): id = 8  
11-22 14:24:35.081: D/myLogs(414): id = 9  
11-22 14:24:35.152: D/myLogs(414): id = 10  
11-22 14:24:35.352: D/myLogs(414): id = 11  
11-22 14:24:35.382: D/myLogs(414): id = 12  
11-22 14:24:35.452: D/myLogs(414): id = 13  
11-22 14:24:35.532: D/myLogs(414): id = 14  
11-22 14:24:35.621: D/myLogs(414): id = 15  
11-22 14:24:36.272: D/myLogs(414): --- Все записи ---  
11-22 14:24:36.321: D/myLogs(414): id = 1; name = Кембридж; student = 140; country = Англия;  
11-22 14:24:36.321: D/myLogs(414): id = 2; name = Тюбеген; student = 211; country = Германия;  
11-22 14:24:36.332: D/myLogs(414): id = 3; name = Оксфорд; student = 195; country = Англия;  
11-22 14:24:36.342: D/myLogs(414): id = 4; name = Гейдельбург; student = 142; country = Германия;  
11-22 14:24:36.352: D/myLogs(414): id = 5; name = Мюнхен; student = 128; country = Германия;  
11-22 14:24:36.382: D/myLogs(414): id = 6; name = Гарвард; student = 82; country = США;  
11-22 14:24:36.382: D/myLogs(414): id = 7; name = МТИ; student = 280; country = США;  
11-22 14:24:36.382: D/myLogs(414): id = 8; name = Сорбона; student = 60; country = Франция;  
11-22 14:24:36.392: D/myLogs(414): id = 9; name = Принстон; student = 66; country = США;  
11-22 14:24:36.411: D/myLogs(414): id = 10; name = Имперский колледж; student = 35; country = Англия;  
11-22 14:24:36.411: D/myLogs(414): id = 11; name = Страсбург; student = 223; country = Франция;  
11-22 14:24:36.424: D/myLogs(414): id = 12; name = Беркли; student = 56; country = США;  
11-22 14:24:36.432: D/myLogs(414): id = 13; name = МГТУ; student = 110; country = Россия;  
11-22 14:24:36.432: D/myLogs(414): id = 14; name = МГУ; student = 45; country = Россия;  
11-22 14:24:36.462: D/myLogs(414): id = 15; name = ТГУ; student = 121; country = Россия;  
11-22 15:17:09.482: D/myLogs(456): --- Функция count (\*) as Count ---  
11-22 15:17:09.502: D/myLogs(456): Count = 15;  
11-22 15:18:08.312: D/myLogs(456): --- Функция sum(student) as student ---  
11-22 15:18:08.322: D/myLogs(456): student = 1894;  
11-22 15:18:28.722: D/myLogs(456): --- Функция max(student) as student ---  
11-22 15:18:28.742: D/myLogs(456): student = 280;  
11-22 15:18:41.252: D/myLogs(456): --- Функция min(student) as student ---  
11-22 15:18:41.304: D/myLogs(456): student = 35;

11-22 15:20:30.922: D/myLogs(518): --- Студентов больше 180 ---  
11-22 15:20:30.972: D/myLogs(518): id = 2; name = Тюберген; student = 211; country = Германия;  
11-22 15:20:30.972: D/myLogs(518): id = 3; name = Оксфорд; student = 195; country = Англия;  
11-22 15:20:30.981: D/myLogs(518): id = 7; name = МТИ; student = 280; country = США;  
11-22 15:20:30.991: D/myLogs(518): id = 11; name = Страсбург; student = 223; country = Франция;  
11-22 15:20:33.862: D/myLogs(518): --- Студентов в стране ---  
11-22 15:20:33.912: D/myLogs(518): country = Англия; student = 370;  
11-22 15:20:33.912: D/myLogs(518): country = Германия; student = 481;  
11-22 15:20:33.922: D/myLogs(518): country = Россия; student = 276;  
11-22 15:20:33.922: D/myLogs(518): country = США; student = 484;  
11-22 15:20:33.931: D/myLogs(518): country = Франция; student = 283;  
11-22 15:20:43.872: D/myLogs(518): --- Страны с числом студентов больше 280 ---  
11-22 15:20:43.926: D/myLogs(518): country = Англия; student = 370;  
11-22 15:20:43.926: D/myLogs(518): country = Германия; student = 481;  
11-22 15:20:43.932: D/myLogs(518): country = США; student = 484;  
11-22 15:20:43.952: D/myLogs(518): country = Франция; student = 283;  
11-22 15:20:49.762: D/myLogs(518): --- Сортировка по названию вуза ---  
11-22 15:20:49.815: D/myLogs(518): id = 12; name = Беркли; student = 56; country = США;  
11-22 15:20:49.822: D/myLogs(518): id = 6; name = Гарвард; student = 82; country = США;  
11-22 15:20:49.833: D/myLogs(518): id = 4; name = Гейдельбург; student = 142; country = Германия;  
11-22 15:20:49.833: D/myLogs(518): id = 10; name = Имперский колледж; student = 35; country = Англия;  
11-22 15:20:49.842: D/myLogs(518): id = 1; name = Кембридж; student = 140; country = Англия;  
11-22 15:20:49.882: D/myLogs(518): id = 13; name = МГТУ; student = 110; country = Россия;  
11-22 15:20:49.882: D/myLogs(518): id = 14; name = МГУ; student = 45; country = Россия;  
11-22 15:20:49.882: D/myLogs(518): id = 7; name = МТИ; student = 280; country = США;  
11-22 15:20:49.892: D/myLogs(518): id = 5; name = Мюнхен; student = 128; country = Германия;  
11-22 15:20:49.892: D/myLogs(518): id = 3; name = Оксфорд; student = 195; country = Англия;  
11-22 15:20:49.937: D/myLogs(518): id = 9; name = Принстон; student = 66; country = США;  
11-22 15:20:49.937: D/myLogs(518): id = 8; name = Сорбона; student = 60; country = Франция;  
11-22 15:20:49.942: D/myLogs(518): id = 11; name = Страсбург; student = 223; country = Франция;  
11-22 15:20:49.942: D/myLogs(518): id = 15; name = ТГУ; student = 121; country = Россия;  
11-22 15:20:49.992: D/myLogs(518): id = 2; name = Тюберген; student = 211; country = Германия;

11-22 15:20:51.933: D/myLogs(518): --- Сортировка по студентам ---  
11-22 15:20:51.962: D/myLogs(518): id = 10; name = Имперский колледж; student = 35; country = Англия;  
11-22 15:20:51.962: D/myLogs(518): id = 14; name = МГУ; student = 45; country = Россия;  
11-22 15:20:51.991: D/myLogs(518): id = 12; name = Беркли; student = 56; country = США;  
11-22 15:20:52.002: D/myLogs(518): id = 8; name = Сорбона; student = 60; country = Франция;  
11-22 15:20:52.002: D/myLogs(518): id = 9; name = Принстон; student = 66; country = США;  
11-22 15:20:52.002: D/myLogs(518): id = 6; name = Гарвард; student = 82; country = США;  
11-22 15:20:52.062: D/myLogs(518): id = 13; name = МГТУ; student = 110; country = Россия;  
11-22 15:20:52.062: D/myLogs(518): id = 15; name = ТГУ; student = 121; country = Россия;  
11-22 15:20:52.062: D/myLogs(518): id = 5; name = Мюнхен; student = 128; country = Германия;  
11-22 15:20:52.062: D/myLogs(518): id = 1; name = Кембридж; student = 140; country = Англия;  
11-22 15:20:52.072: D/myLogs(518): id = 4; name = Гейдельбург; student = 142; country = Германия;  
11-22 15:20:52.072: D/myLogs(518): id = 3; name = Оксфорд; student = 195; country = Англия;  
11-22 15:20:52.072: D/myLogs(518): id = 2; name = Тюбеген; student = 211; country = Германия;  
11-22 15:20:52.172: D/myLogs(518): id = 11; name = Страсбург; student = 223; country = Франция;  
11-22 15:20:52.182: D/myLogs(518): id = 7; name = МТИ; student = 280; country = США;  
11-22 15:20:53.752: D/myLogs(518): --- Сортировка по стране ---  
11-22 15:20:53.773: D/myLogs(518): id = 1; name = Кембридж; student = 140; country = Англия;  
11-22 15:20:53.804: D/myLogs(518): id = 3; name = Оксфорд; student = 195; country = Англия;  
11-22 15:20:53.822: D/myLogs(518): id = 10; name = Имперский колледж; student = 35; country = Англия;  
11-22 15:20:53.862: D/myLogs(518): id = 2; name = Тюбеген; student = 211; country = Германия;  
11-22 15:20:53.862: D/myLogs(518): id = 4; name = Гейдельбург; student = 142; country = Германия;  
11-22 15:20:53.912: D/myLogs(518): id = 5; name = Мюнхен; student = 128; country = Германия;  
11-22 15:20:53.912: D/myLogs(518): id = 13; name = МГТУ; student = 110; country = Россия;  
11-22 15:20:53.922: D/myLogs(518): id = 14; name = МГУ; student = 45; country = Россия;  
11-22 15:20:53.967: D/myLogs(518): id = 15; name = ТГУ; student = 121; country = Россия;  
11-22 15:20:53.967: D/myLogs(518): id = 6; name = Гарвард; student = 82; country = США;  
11-22 15:20:53.972: D/myLogs(518): id = 7; name = МТИ; student = 280; country = США;  
11-22 15:20:53.972: D/myLogs(518): id = 9; name = Принстон; student = 66; country = США;  
11-22 15:20:53.982: D/myLogs(518): id = 12; name = Беркли; student = 56; country = США;  
11-22 15:20:53.982: D/myLogs(518): id = 8; name = Сорбона; student = 60; country = Франция;  
11-22 15:20:53.992: D/myLogs(518): id = 11; name = Страсбург; student = 223; country = Франция;

# Запросы из связанных таблиц. INNER JOIN в SQLite.

## Метод rawQuery

Рассмотрим, как с помощью метода query выполнять запросы для связанных таблиц. Создадим простое приложение, которое будет делать запрос из двух таблиц и выводить результат в лог.

Таблицы будут people и position. В первую (people) запишем список людей, во вторую (position) – список должностей. И для каждого человека в people будет прописан id должности из position.

Экран использоваться не будет, поэтому activity\_main.xml остается как есть.

## //Java

```
public class MainActivity extends Activity {
    final String LOG_TAG = "myLogs";
    // данные для таблицы должностей
    int[] position_id = { 1, 2, 3, 4 };
    String[] position_name = { "Директор", "Программист", "Бухгалтер", "Охранник" };
    int[] position_salary = { 15000, 13000, 10000, 8000 };
    // данные для таблицы людей
    String[] people_name = { "Иван", "Марья", "Петр", "Антон", "Даша", "Борис", "Костя", "Игорь" };
    int[] people_posid = { 2, 3, 2, 2, 3, 1, 2, 4 };
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // Подключаемся к БД
        DBHelper dbh = new DBHelper(this);
        SQLiteDatabase db = dbh.getWritableDatabase();
        // Описание курсора
        Cursor c;
        // выводим в лог данные по должностям
        Log.d(LOG_TAG, "---- Table position ----");
        c = db.query("position", null, null, null, null, null, null);
        logCursor(c);
        Log.d(LOG_TAG, "---- ----");
        // выводим в лог данные по людям
        Log.d(LOG_TAG, "---- Table people ----");
        c = db.query("people", null, null, null, null, null, null);
        logCursor(c);
        Log.d(LOG_TAG, "---- ----");
        // выводим результат объединения
        // используем rawQuery
        Log.d(LOG_TAG, "---- INNER JOIN with rawQuery----");
        String sqlQuery = "select PL.name as Name, PS.name as Position, salary as Salary "
            + "from people as PL "
            + "inner join position as PS "
            + "on PL.posid = PS.id "
            + "where salary > ?";
        c = db.rawQuery(sqlQuery, new String[] { "12000" });
        logCursor(c);
        Log.d(LOG_TAG, "---- ----");
        // выводим результат объединения
        // используем query
        Log.d(LOG_TAG, "---- INNER JOIN with query----");
        String table = "people as PL inner join position as PS on PL.posid = PS.id";
        String columns[] = { "PL.name as Name", "PS.name as Position", "salary as Salary" };
        String selection = "salary < ?";
        String[] selectionArgs = { "12000" };
        c = db.query(table, columns, selection, selectionArgs, null, null, null);
        logCursor(c);
        Log.d(LOG_TAG, "---- ----");
        // закрываем БД
        dbh.close();
    }
}
```

```

// вывод в лог данных из курсора
void logCursor(Cursor c) {
    if (c != null) {
        if (c.moveToFirst()) {
            String str;
            do {
                str = "";
                for (String cn : c.getColumnNames()) {
                    str = str.concat(cn + " = " +
                        c.getString(c.getColumnIndex(cn)) + "; ");
                }
                Log.d(LOG_TAG, str);
            } while (c.moveToNext());
        } else
            Log.d(LOG_TAG, "Cursor is null");
    }
    // класс для работы с БД
    class DBHelper extends SQLiteOpenHelper {

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

```

```

public void onCreate(SQLiteDatabase db) {
    Log.d(LOG_TAG, "---- onCreate database ----");
    ContentValues cv = new ContentValues();
    // создаем таблицу должностей
    db.execSQL("create table position (" + "id integer primary key," +
        "name text," + "salary integer" + ");");
    // заполняем ее
    for (int i = 0; i < position_id.length; i++) {
        cv.clear();
        cv.put("id", position_id[i]);
        cv.put("name", position_name[i]);
        cv.put("salary", position_salary[i]);
        db.insert("position", null, cv);
    }
    // создаем таблицу людей
    db.execSQL("create table people (" + "id integer primary key" +
        "autoincrement," + "name text," + "posid integer" + ");");
    // заполняем ее
    for (int i = 0; i < people_name.length; i++) {
        cv.clear();
        cv.put("name", people_name[i]);
        cv.put("posid", people_posid[i]);
        db.insert("people", null, cv);
    }
}

public void onUpgrade(SQLiteDatabase db, int oldVersion, int
    newVersion) {
}
}
}

```

# //Kotlin

```
class MainActivity : AppCompatActivity() {

    val LOG_TAG = "myLogs";
    // данные для таблицы должностей
    var position_id = arrayOf(1, 2, 3, 4)
    var position_name = arrayOf("Директор", "Программист", "Бухгалтер", "Охранник")
    var position_salary = arrayOf(15000, 13000, 10000, 8000)
    // данные для таблицы людей
    var people_name = arrayOf("Иван", "Марья", "Петр", "Антон", "Даша", "Борис", "Костя", "Игорь")
    var people_posid = arrayOf(2, 3, 2, 2, 3, 1, 2, 4)

    val SQL_CREATE_TABLE1 = "create table IF NOT EXISTS position (id integer primary key," +
        " name text, salary integer)"
    val SQL_CREATE_TABLE2 = "create table IF NOT EXISTS people (id integer primary key " +
        "autoincrement, name text, posid integer);"
    private var db: SQLiteDatabase? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        Log.d(LOG_TAG, "open")
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        db = openOrCreateDatabase("myDB", Context.MODE_PRIVATE, null)
        db!!.execSQL(SQL_CREATE_TABLE1)
        var c = db!!.query("position", null, null, null,
            null, null, null)
        if (c.count == 0) {
            var cv = ContentValues()
            for (i in 0..3) {
                cv.clear()
                cv.put("id", position_id[i])
                cv.put("name", position_name[i])
                cv.put("salary", position_salary[i])
                db!!.insert("position", null, cv)
            }
        }
    }
}
```

```

db!!.execSQL(SQL_CREATE_TABLE2)
c = db!!.query("people", null, null, null,
    null, null, null)
if (c.count == 0) {
    var cv = ContentValues()
    for (i in 0..7) {
        cv.clear()
        cv.put("name", people_name[i])
        cv.put("posid", people_posid[i])
        db!!.insert("people", null, cv)
    }
}

```

```

Log.d(LOG_TAG, "--- Table position ---")
c = db!!.query("position", null, null, null, null, null, null)
logCursor(c)
Log.d(LOG_TAG, "--- ---")
// выводим в лог данные по людям
Log.d(LOG_TAG, "--- Table people ---")
c = db!!.query("people", null, null, null, null, null, null)
logCursor(c)
Log.d(LOG_TAG, "--- ---")
// выводим результат объединения
// используем rawQuery
Log.d(LOG_TAG, "--- INNER JOIN with rawQuery---")
val sqlQuery = ("select PL.name as Name, PS.name as Position, salary as Salary "
    + "from people as PL "
    + "inner join position as PS "
    + "on PL.posid = PS.id "
    + "where salary > ?")
c = db!!.rawQuery(sqlQuery, arrayOf("12000"))
logCursor(c)
Log.d(LOG_TAG, "--- ---")

```



```

// выводим результат объединения
// используем query
Log.d(LOG_TAG, "--- INNER JOIN with query---")
val table = "people as PL inner join position as PS on PL.posid = PS.id"
val columns = arrayOf("PL.name as Name", "PS.name as Position", "salary as
Salary")
val selection = "salary < ?"
val selectionArgs = arrayOf("12000")
c = db!!.query(table, columns, selection, selectionArgs, null, null, null)
logCursor(c)
Log.d(LOG_TAG, "--- ---")
}

fun logCursor(c: Cursor){
    if (c != null) {
        if (c.moveToFirst()) {
            var str: String
            do {
                str = ""
                for (cn in c.columnNames) {
                    str = str + (cn + " = " + c.getString(c.getColumnIndex(cn)) + ";
")

                }
                Log.d(LOG_TAG, str)
            } while (c.moveToNext())
        }
    } else
        Log.d(LOG_TAG, "Cursor is null")
}
}

```

Сначала идут несколько массивов с данными для таблиц. Обратите внимание, для должностей id указывается при заполнении таблиц. Это сделано для того, чтобы знать эти номера и использовать их в таблице людей для указания id должности.

В методе Activity onCreate создается объект для управления БД и подключение к БД. Далее используя query выводятся в лог данные из таблиц position и people.

Для вывода объединения таблиц используется rawQuery. Это несложный метод, который принимает на вход SQL-запрос и список аргументов для условия WHERE (если необходимо). В приложении сформирован запрос на объединение двух таблиц и вывода имени, должности и зарплаты человека. Условие выборки: ЗП должна быть больше 12000. Для формирования условия используются аргументы.

Далее снова выводится объединение таблиц, но используется обычный query. В table записываются все таблицы, их алиасы и условие JOIN. В columns – все нужные поля с использованием алиасов. В selection и selectionArgs записано условие выборки – ЗП меньше 12000.

Метод logCursor получает на вход Cursor и выводит в лог все содержимое.

11-22 15:24:15.612: D/myLogs(555): --- Table position ---  
11-22 15:24:15.641: D/myLogs(555): id = 1; name = Директор; salary = 15000;  
11-22 15:24:15.641: D/myLogs(555): id = 2; name = Программист; salary = 13000;  
11-22 15:24:15.652: D/myLogs(555): id = 3; name = Бухгалтер; salary = 10000;  
11-22 15:24:15.671: D/myLogs(555): id = 4; name = Охранник; salary = 8000;  
11-22 15:24:15.681: D/myLogs(555): --- ---  
11-22 15:24:15.681: D/myLogs(555): --- Table people ---  
11-22 15:24:15.702: D/myLogs(555): id = 1; name = Иван; posid = 2;  
11-22 15:24:15.702: D/myLogs(555): id = 2; name = Марья; posid = 3;  
11-22 15:24:15.732: D/myLogs(555): id = 3; name = Петр; posid = 2;  
11-22 15:24:15.742: D/myLogs(555): id = 4; name = Антон; posid = 2;  
11-22 15:24:15.752: D/myLogs(555): id = 5; name = Даша; posid = 3;  
11-22 15:24:15.752: D/myLogs(555): id = 6; name = Борис; posid = 1;  
11-22 15:24:15.752: D/myLogs(555): id = 7; name = Костя; posid = 2;  
11-22 15:24:15.762: D/myLogs(555): id = 8; name = Игорь; posid = 4;  
11-22 15:24:15.762: D/myLogs(555): --- ---  
11-22 15:24:15.762: D/myLogs(555): --- INNER JOIN with rawQuery---  
11-22 15:24:15.811: D/myLogs(555): Name = Иван; Position = Программист; Salary = 13000;  
11-22 15:24:15.811: D/myLogs(555): Name = Петр; Position = Программист; Salary = 13000;  
11-22 15:24:15.872: D/myLogs(555): Name = Антон; Position = Программист; Salary = 13000;  
11-22 15:24:15.883: D/myLogs(555): Name = Борис; Position = Директор; Salary = 15000;  
11-22 15:24:15.922: D/myLogs(555): Name = Костя; Position = Программист; Salary = 13000;  
11-22 15:24:15.922: D/myLogs(555): --- ---  
11-22 15:24:15.922: D/myLogs(555): --- INNER JOIN with query---  
11-22 15:24:15.931: D/myLogs(555): Name = Марья; Position = Бухгалтер; Salary = 10000;  
11-22 15:24:15.931: D/myLogs(555): Name = Даша; Position = Бухгалтер; Salary = 10000;  
11-22 15:24:15.942: D/myLogs(555): Name = Игорь; Position = Охранник; Salary = 8000;  
11-22 15:24:15.942: D/myLogs(555): --- ---