[Запросы из связанных таблиц. INNER JOIN в SQLite. Метод rawQuery.](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/77-urok-37-zaprosy-iz-svjazannyh-tablits-inner-join-v-sqlite-metod-rawquery.html" \o "Урок 37. Запросы из связанных таблиц. INNER JOIN в SQLite. Метод rawQuery.)

Мы достаточно подробно разобрали метод **query**. Но не рассмотрели, как с его помощью 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**.

Создадим проект:

**Project name:** P0371\_SQLiteInnerJoin  
**Build Target:** Android  
**Application name:** SQLiteInnerJoin  
**Package name:** ru.startandroid.develop.p0371sqliteinnerjoin  
**Create Activity:** MainActivity

Экран вообще использовать не будем, поэтому **main.xml** даже не трогаем. Открываем **MainActivity.java** и пишем код:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145 | package ru.startandroid.develop.p0371sqliteinnerjoin;    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;    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", "Программер", "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 };      /\*\* Called when the activity is first created. \*/    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);      c.close();      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);      c.close();      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);      c.close();      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);      c.close();      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) {        }    }    } |

Разбираем код. Сначала идут несколько **массивов**с **данными**для таблиц. Обратите внимание, для **должностей**мы будем указывать **id**при заполнении таблиц. Это сделано для того, чтобы мы знали эти номера и могли их использовать в таблице**людей**для указания id должности.

В методе Activity **onCreate**мы создаем объект для **управления**БД и **подключаемся**к БД. Далее используя знакомый нам**query**выводим в лог данные из таблиц **position**и **people**.

Для вывода объединения таблиц используем [rawQuery](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "rawQuery(java.lang.String, java.lang.String[])" \t "_blank). Это несложный метод, который принимает на вход **SQL-запрос** и список **аргументов**для условия WHERE (если необходимо). Мы сформировали запрос на объединение двух таблиц и вывода**имени**, **должности**и **зарплаты**человека. Условие выборки: **ЗП**должна быть **больше 12000**. Мы использовали аргументы для формирования условия.

Далее снова выводим объединение таблиц, но используем обычный **query**. В **table**записываем все таблицы, их алиасы и условие **JOIN**. В **columns**– все нужные поля с использованием алиасов. Ну и в **selection** и **selectionArgs**пишем условие выборки – **ЗП меньше 12000**.

Наш метод **logCursor** получает на вход **Cursor**и выводит в лог все **содержимое**. Тут все знакомо с прошлых уроков.

В классе **DBHelper**кодим **создание** таблиц и, на этот раз, здесь же их **наполнение**данными.

Все сохраним и запустим. Смотрим лог:

*--- onCreate database ---  
--- Table position ---  
id = 1; name = Директор; salary = 15000;  
id = 2; name = Программер; salary = 13000;  
id = 3; name = Бухгалтер; salary = 10000;  
id = 4; name = Охранник; salary = 8000;  
--- ---  
--- Table people ---  
id = 1; name = Иван; posid = 2;  
id = 2; name = Марья; posid = 3;  
id = 3; name = Петр; posid = 2;  
id = 4; name = Антон; posid = 2;  
id = 5; name = Даша; posid = 3;  
id = 6; name = Борис; posid = 1;  
id = 7; name = Костя; posid = 2;  
id = 8; name = Игорь; posid = 4;  
--- ---*

Вывели данные по таблицам раздельно.

*--- INNER JOIN with rawQuery---  
Name = Иван; Position = Программер; Salary = 13000;  
Name = Петр; Position = Программер; Salary = 13000;  
Name = Антон; Position = Программер; Salary = 13000;  
Name = Борис; Position = Директор; Salary = 15000;  
Name = Костя; Position = Программер; Salary = 13000;  
--- ---*

Вывели данные из rawQuery. ЗП > 12000

*--- INNER JOIN with query---  
Name = Марья; Position = Бухгалтер; Salary = 10000;  
Name = Даша; Position = Бухгалтер; Salary = 10000;  
Name = Игорь; Position = Охранник; Salary = 8000;  
--- ---*

Вывели данные из query. ЗП < 12000

Как видите, запросы из связанных таблиц в **SQLite**– не проблема и ничем не отличается от обычных БД.

Что использовать **rawQuery**или **query**– решать вам и зависит от ситуации. Хотя, навскидку я не могу придумать преимущества одного над другим в той или иной ситуации. Но наверно они есть.

Данные по ЗП и должностям являются выдуманными, любое совпадение – случайно. И, разумеется, ничего против бухгалтеров и охранников я не имею ))

На следующем уроке:

- используем транзакции при работе с БД

Похожие статьи

* [Урок 136. CursorLoader](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/278-urok-136-cursorloader.html)
* [Урок 101. Создаем свой ContentProvider](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/166-urok-101-sozdaem-svoj-contentprovider.html)
* [Урок 53. SimpleCursorTreeAdapter, пример использования](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/112-urok-53-simplecursortreeadapter-primer-ispolzovanija.html)
* [Урок 52. SimpleCursorAdapter, пример использования](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/111-urok-52-simplecursoradapter-primer-ispolzovanija.html)
* [Урок 39. onUpgrade. Обновляем БД в SQLite](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/79-urok-39-onupgrade-obnovljaem-bd-v-sqlite.html)
* [Урок 38. Транзакции в SQLite. Небольшой FAQ по SQLite.](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/78-urok-38-tranzaktsii-v-sqlite.html)
* [Урок 36. SQLite. Подробнее про метод query. Условие, сортировка, группировка](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/76-urok-36-sqlite-podrobnee-pro-metod-query-uslovie-sortirovka-gruppirovka.html)
* [Урок 35. SQLite. Методы update и delete с указанием условия](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/75-urok-35-metody-query-i-delete-s-ukazaniem-uslovija.html)
* [Урок 34. Хранение данных. SQLite](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/74-urok-34-hranenie-dannyh-sqlite.html)

Последние статьи

* [Урок 176. OpenGL. Индексы, текстуры для куба.](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/408-urok-176-opengl-indeksy-tekstury-dlja-kuba.html)
* [Урок 175. OpenGL. Текстуры.](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/406-urok-175-opengl-tekstury.html)
* [Урок 174. OpenGL. Модель](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/405-urok-174-opengl-model.html)
* [Урок 173. OpenGL. Камера](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/403-urok-173-opengl-kamera.html)
* [Урок 172. OpenGL. Perspective. Frustum. Ortho.](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/401-urok-172-perspective-frustum-ortho.html)