Запросы из связанных таблиц. INNER JOIN в SQLite.

Метод rawQuery.

В этом уроке: - читаем данные из связанных таблиц - используем 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 списоклюдей, 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 2.3.3 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:

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. 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: ЗПдолжна 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  
  
Источник: <https://intellect.ml/37-zaprosy-iz-svyazannykh-tablits-inner-join-v-sqlite-metod-rawquery-5879>