[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" \o "Урок 35. SQLite. Методы update и delete с указанием условия)

На прошлом уроке мы разобрали, как **вставить**запись, **считать**все записи из таблицы и **очистить**таблицу. Теперь посмотрим, как **обновить**и **удалить**конкретную запись.

Новый проект создавать не будем, используем **P0341\_SimpleSQLite** с прошлого урока. Немного поменяем экран, добавим **поле** для ввода **ID** и **кнопки**для **обновления**и **удаления**.

Перепишем **main.xml**:

<?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:text="ID"

            android:layout\_marginLeft="5dp"

            android:layout\_marginRight="25dp">

        </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:text="Name"

            android:layout\_marginLeft="5dp"

            android:layout\_marginRight="5dp">

        </TextView>

        <EditText

            android:id="@+id/etName"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:layout\_weight="1">

            <requestFocus>

            </requestFocus>

        </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:text="Email"

            android:layout\_marginLeft="5dp"

            android:layout\_marginRight="5dp">

        </TextView>

        <EditText

            android:id="@+id/etEmail"

            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>

По нажатию кнопки **Update**мы будем читать содержимое полей **Name**и **Email**, и обновлять запись в таблице, для которой **id** = значению из поля **ID**. По нажатию кнопки **Delete**будем удалять запись из таблицы по **id**= значению из поля **ID**. Экран получился, конечно, не самый лучший с точки зрения эргономики и юзабилити, но у нас тут не кружок юных дизайнеров, а серьезное изучение БД.

Подредактируем **MainActivity.java**.Добавим описание и определение новых экранных элементов, присвоение обработчиков для кнопок.

final String LOG\_TAG = "myLogs";

Button btnAdd, btnRead, btnClear, btnUpd, btnDel;

EditText etName, etEmail, etID;

...

public void onCreate(Bundle savedInstanceState) {

  ...

  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);

  etEmail = (EditText) findViewById(R.id.etEmail);

  etID = (EditText) findViewById(R.id.etID);

  // создаем объект для создания и управления версиями БД

  dbHelper = new DBHelper(this);

}

(Обращайте внимание только на выделенные строки)

Теперь дополним реализацию **onClick**:

public void onClick(View v) {

   // создаем объект для данных

   ContentValues cv = new ContentValues();

   // получаем данные из полей ввода

   String name = etName.getText().toString();

   String email = etEmail.getText().toString();

   String id = etID.getText().toString();

   // подключаемся к БД

   SQLiteDatabase db = dbHelper.getWritableDatabase();

   switch (v.getId()) {

   case R.id.btnAdd:

     ...

   case R.id.btnRead:

     ...

   case R.id.btnClear:

     ...

   case R.id.btnUpd:

     if (id.equalsIgnoreCase("")) {

       break;

     }

     Log.d(LOG\_TAG, "--- Update mytable: ---");

     // подготовим значения для обновления

     cv.put("name", name);

     cv.put("email", email);

     // 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 mytable: ---");

     // 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();

 }

(Добавляете только выделенный текст)

Мы добавляем переменную **id**, пишем в нее значение поля **etID**. В **switch**добавляем две новые ветки:

**btnUpd**– **обновление**записи в mytable. Проверяем, что значение **id**не пустое, заполняем **cv**данными для апдейта и**обновляем**запись. Для этого используется метод **update**. На вход ему подается **имя** таблицы, заполненный **ContentValues**с значениями для обновления, строка **условия**(Where) и массив **аргументов**для строки условия. В строке условия я использовал знак *?*. При запросе к БД вместо этого знака будет подставлено значение из массива **аргументов**, в нашем случае это – значение переменной **id**. Если знаков *?* в строке условия несколько, то им будут сопоставлены значения из массива по порядку. Метод **update**возвращает нам **кол-во обновленных** записей, которое мы выводим в лог.

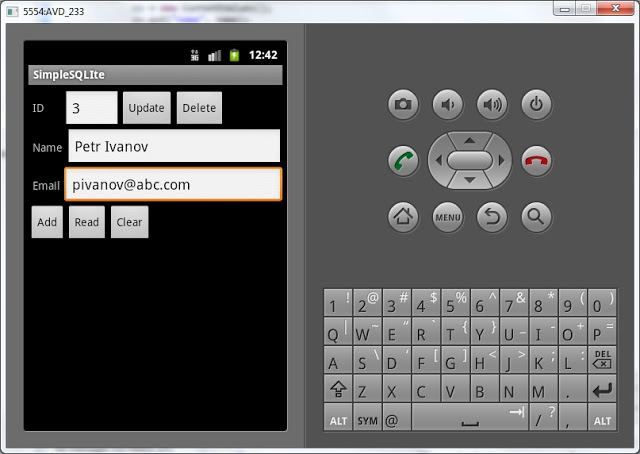
**btnDel**– **удаление**записи из mytable. Проверяем, что **id**не пустое и вызываем метод **delete**. На вход передаем **имя**таблицы, строку **условия**и массив **аргументов**для условия. Метод delete возвращает кол-во удаленных строк, которое мы выводим в лог.

Обратите внимание, что **условия**и для **update**и для **delete**у меня одинаковые, а именно ***id****= значение из поля****etID***. Но реализовал я их немного по-разному. Для **update**использовал символ *?* в строке условия и **массив**аргументов. А для **delete**вставил значение сразу в строку **условия**. Таким образом, я просто показал**способы формирования условия**. А вы уже используйте тот, что больше нравится или лучше в конкретной ситуации.

Все сохраним и запустим. На прошлом уроке мы добавляли пару записей, но потом их удалили. Так что я буду добавлять снова. **Добавлю**пару записей, нажму **Read**, вижу в логе:

*ID = 3, name = Ivan Petrov, email = ipetrov @abc.com  
ID = 4, name = Anton Sidorov, email = asidorov @def.com*

Теперь попробуем обновить запись с **ID**=3. Для этого вводим 3 в поле **ID** и новые данные в поля **Name**и **Email**:



Жмем **Update**, смотрим лог:

*-- Update mytable: ---  
updated rows count = 1*

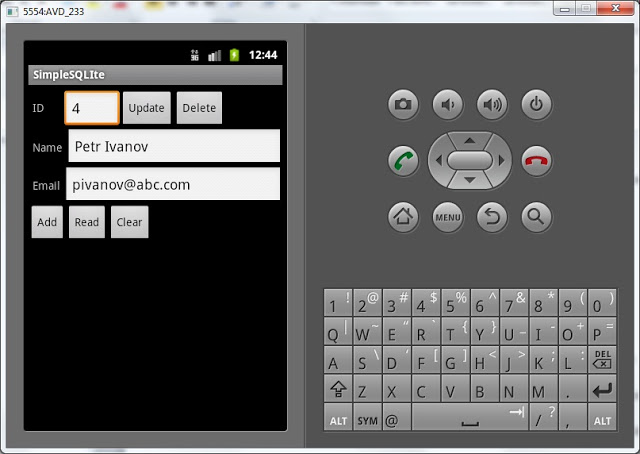
обновилась одна запись, все верно.

Нажмем **Read**и убедимся в этом. Лог:

*ID = 3, name = Petr Ivanov, email = pivanov @abc.com  
ID = 4, name = Anton Sidorov, email = asidorov @def.com*

Запись действительно обновилась.

Теперь давайте удалим запись с **ID**= 4. Вводим 4 в поле **ID**



Жмем **Delete**, смотрим лог:

*--- Delete from mytable: ---  
deleted rows count = 1*

одна запись удалена.

Жмем **Read**, смотрим лог:

*--- Rows in mytable: ---  
ID = 3, name = Petr Ivanov, email = pivanov @abc.com*

осталась одна запись.

Если попробовать удалить запись с **пустым**полем **ID**, то ничего не будет, т.к. мы реализовали проверку. Если же попробовать удалить запись с несуществующим **ID**, то метод **delete**вернет **0**. Т.е. ничего не было удалено, т.к. не нашлось записей для удаления.

Теперь вы умеете **читать**, **вставлять**, **удалять**и **изменять**записи в SQLite.

Полный код **MainActivity.java**:

package ru.startandroid.develop.p0341simplesqlite;

import ru.startandroid.develop.p0341simpelsqlite.R;

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, btnUpd, btnDel;

  EditText etName, etEmail, etID;

  DBHelper dbHelper;

  /\*\* Called when the activity is first created. \*/

  public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.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);

    etEmail = (EditText) findViewById(R.id.etEmail);

    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 email = etEmail.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("email", email);

      // 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 emailColIndex = c.getColumnIndex("email");

        do {

          // получаем значения по номерам столбцов и пишем все в лог

          Log.d(LOG\_TAG,

              "ID = " + c.getInt(idColIndex) + ", name = "

                  + c.getString(nameColIndex) + ", email = "

                  + 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");

      c.close();

      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 mytable: ---");

      // подготовим значения для обновления

      cv.put("name", name);

      cv.put("email", email);

      // 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 mytable: ---");

      // 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,"

          + "email text" + ");");

    }

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

    }

  }

}