**Работа с базой данных SQLite**

В данном уроке мы рассмотрим как можно взаимодействовать с базой данных в Android приложениях.

**Android** предоставляет несколько способов для хранения пользовательских данных и приложений. **SQLite** является одним из способов хранения данных пользователя. SQLite это очень легковестная база данных, которая содержится в **ОС Android**.

[Скачать](https://devcolibri.com/download/357/DBAndroid.zip)

В этом уроке я продемонстрирую, как реализуются запросы для обработки их как SQLite операций.

***Начиная с этого урока я буду выкладывать вместе со статьей, обзор того как должно работать приложение после прохождения урока.***

**Шаг 1.**

Начнем реализацию с класса **DatabaseHandler**, но для начала создадим для этого класса интерфейс:

public interface IDatabaseHandler {

public void addContact(Contact contact);

public Contact getContact(int id);

public List<Contact> getAllContacts();

public int getContactsCount();

public int updateContact(Contact contact);

public void deleteContact(Contact contact);

public void deleteAll();

}

***Описание методов:***

void **addContact**(Contact contact) – позволяет сохранять в базу данных новые контакты пользователей;  
Contact **getContact**(int id) – позволяет получить контакты по **id**;  
List **getAllContacts**() – позволяет получить все контакты с БД;  
int **getContactsCount**() – позволяет получить количество контактов находящиеся в БД;  
int **updateContact**(Contact contact) – позволяет обновить контакт;  
void **deleteContact**(Contact contact) – позволяет удалить контакт;  
void **deleteAll**() – позволяет удалить все контакты.

Теперь нужно реализовать этот интерфейс, но для начала унаследуемся от **SQLiteOpenHelper .**

Полный класс выглядит так:

package com.devcolibri.DBAndroid;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

import java.util.List;

public class DatabaseHandler extends SQLiteOpenHelper implements IDatabaseHandler {

private static final int DATABASE\_VERSION = 1;

private static final String DATABASE\_NAME = "contactsManager";

private static final String TABLE\_CONTACTS = "contacts";

private static final String KEY\_ID = "id";

private static final String KEY\_NAME = "name";

private static final String KEY\_PH\_NO = "phone\_number";

public DatabaseHandler(Context context) {

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

@Override

public void onCreate(SQLiteDatabase db) {

String CREATE\_CONTACTS\_TABLE = "CREATE TABLE " + TABLE\_CONTACTS + "("

+ KEY\_ID + " INTEGER PRIMARY KEY," + KEY\_NAME + " TEXT,"

+ KEY\_PH\_NO + " TEXT" + ")";

db.execSQL(CREATE\_CONTACTS\_TABLE);

}

@Override

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

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_CONTACTS);

onCreate(db);

}

@Override

public void addContact(Contact contact) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(KEY\_NAME, contact.getName());

values.put(KEY\_PH\_NO, contact.getPhoneNumber());

db.insert(TABLE\_CONTACTS, null, values);

db.close();

}

@Override

public Contact getContact(int id) {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.query(TABLE\_CONTACTS, new String[] { KEY\_ID,

KEY\_NAME, KEY\_PH\_NO }, KEY\_ID + "=?",

new String[] { String.valueOf(id) }, null, null, null, null);

if (cursor != null){

cursor.moveToFirst();

}

Contact contact = new Contact(Integer.parseInt(cursor.getString(0)), cursor.getString(1), cursor.getString(2));

return contact;

}

@Override

public List<Contact> getAllContacts() {

List<Contact> contactList = new ArrayList<Contact>();

String selectQuery = "SELECT \* FROM " + TABLE\_CONTACTS;

SQLiteDatabase db = this.getWritableDatabase();

Cursor cursor = db.rawQuery(selectQuery, null);

if (cursor.moveToFirst()) {

do {

Contact contact = new Contact();

contact.setID(Integer.parseInt(cursor.getString(0)));

contact.setName(cursor.getString(1));

contact.setPhoneNumber(cursor.getString(2));

contactList.add(contact);

} while (cursor.moveToNext());

}

return contactList;

}

@Override

public int updateContact(Contact contact) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(KEY\_NAME, contact.getName());

values.put(KEY\_PH\_NO, contact.getPhoneNumber());

return db.update(TABLE\_CONTACTS, values, KEY\_ID + " = ?",

new String[] { String.valueOf(contact.getID()) });

}

@Override

public void deleteContact(Contact contact) {

SQLiteDatabase db = this.getWritableDatabase();

db.delete(TABLE\_CONTACTS, KEY\_ID + " = ?", new String[] { String.valueOf(contact.getID()) });

db.close();

}

@Override

public void deleteAll() {

SQLiteDatabase db = this.getWritableDatabase();

db.delete(TABLE\_CONTACTS, null, null);

db.close();

}

@Override

public int getContactsCount() {

String countQuery = "SELECT \* FROM " + TABLE\_CONTACTS;

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery(countQuery, null);

cursor.close();

return cursor.getCount();

}

}

По названиям переменных класса можно понять зачем они по этому перейду к методам класса.

public DatabaseHandler(Context context) {

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

В конструкторе мы вызываем конструктор предка и передаем название БД, а также версию.

@Override

public void onCreate(SQLiteDatabase db) {

String CREATE\_CONTACTS\_TABLE = "CREATE TABLE " + TABLE\_CONTACTS + "("

+ KEY\_ID + " INTEGER PRIMARY KEY," + KEY\_NAME + " TEXT,"

+ KEY\_PH\_NO + " TEXT" + ")";

db.execSQL(CREATE\_CONTACTS\_TABLE);

}

Этот метод будет создавать таблицу в БД.

@Override

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

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_CONTACTS);

onCreate(db);

}

Этот метод пересоздаст таблицу в БД.

Теперь рассмотрим один метод который добавляет запись в БД, остальные работают аналогичным способом.

@Override

public void addContact(Contact contact) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(KEY\_NAME, contact.getName());

values.put(KEY\_PH\_NO, contact.getPhoneNumber());

db.insert(TABLE\_CONTACTS, null, values);

db.close();

}

В **3-й** строчке примера выше используется экземпляр класса который позволяет нам получать доступ к БД.

**4-я** строчка используется **ContentValues**этот класс содержит данные которые нужно записать в БД в виде **Map** которая принимает два параметра **ключ, значение**с помощью метода **put**(key, value).

Ну и 8-я строка добавляет в запись в БД и следующей строкой закрывает соединение с БД.

**Шаг 2.**

Создаем сущность **Contact:**

package com.devcolibri.DBAndroid;

public class Contact {

int \_id;

String \_name;

String \_phone\_number;

public Contact(){

}

public Contact(int id, String name, String \_phone\_number){

this.\_id = id;

this.\_name = name;

this.\_phone\_number = \_phone\_number;

}

public Contact(String name, String \_phone\_number){

this.\_name = name;

this.\_phone\_number = \_phone\_number;

}

public int getID(){

return this.\_id;

}

public void setID(int id){

this.\_id = id;

}

public String getName(){

return this.\_name;

}

public void setName(String name){

this.\_name = name;

}

public String getPhoneNumber(){

return this.\_phone\_number;

}

public void setPhoneNumber(String phone\_number){

this.\_phone\_number = phone\_number;

}

}

**Шаг 3.**

Пишем **MainActivity:**

package com.devcolibri.DBAndroid;

import android.app.Activity;

import android.os.Bundle;

import java.util.List;

public class Main extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

DatabaseHandler db = new DatabaseHandler(this);

System.out.println("Inserting ..");

db.addContact(new Contact("Empty and One contact", "806800000"));

System.out.println("Reading all contacts..");

List<Contact> contacts = db.getAllContacts();

for (Contact cn : contacts) {

String log = "Id: "+cn.getID()+" ,Name: " + cn.getName() + " ,Phone: " + cn.getPhoneNumber();

System.out.print("Name: ");

System.out.println(log);

}

db.deleteAll();

}

}

**Запускаем!**