Простейшая база данных Часть вторая.

Настоятельно рекомендую разобраться с [предыдущим примером](http://developer.alexanderklimov.ru/android/database.php), чтобы понять основы. После этого можно попробовать написать более сложный пример, где можно работать с записями базы данных. Возьмём проект из прошлого примера и начнём его дорабатывать.

Создадим новый класс **Contact**, который будет содержать сведения о контакте, в нашем случае, сведения о котике.

package ru.alexanderklimov.database;

public class Contact {

private int mId;

private String mName;

private String mPhoneNumber;

private int mAge;

// Пустой констуктор

public Contact() {

}

// Конструктор с параметрами

public Contact(int id, String name, String phoneNumber, int age) {

this.mId = id;

this.mName = name;

this.mPhoneNumber = phoneNumber;

this.mAge = age;

}

// Конструктор с параметрами

public Contact(String name, String phonenumber, int age) {

this.mName = name;

this.mPhoneNumber = phonenumber;

this.mAge = age;

}

// Создание геттеров-сеттеров

public int getID() {

return this.mId;

}

public void setID(int id) {

this.mId = id;

}

public String getName() {

return this.mName;

}

public void setName(String name) {

this.mName = name;

}

public String getPhoneNumber() {

return this.mPhoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.mPhoneNumber = phoneNumber;

}

public int getAge() {

return this.mAge;

}

public void setAge(int age) {

this.mAge = age;

}

}

Теперь вернёмся к классу **DatabaseHelper** и доработаем его, чтобы мы могли выполнять операции CRUD (Create, Read, Update и Delete). Добавим заготовки для методов.

// Добавляем новый контакт

public void addContact(Contact contact) {}

// Получить контакт по идентификатору

public Contact getContact(int id) {}

// Получить все контакты

public List<Contact> getAllContacts() {}

// Получить число контактов

public int getContactsCount() {}

// Обновить контакт

public int updateContact(Contact contact) {}

// Удалить контакт

public void deleteContact(Contact contact) {}

Небольшое отступление. Так как я часто пользуюсь автозаполнением и подсказками, то обнаружил, что константы для колонок таблицы удобнее начинать со слова **COLUMN\_**, а не помещать его в конец. Так быстрее находится нужная константа.

// Рефакторинг имён для констант

public static final String COLUMN\_CAT\_NAME = "cat\_name";

public static final String COLUMN\_PHONE = "phone";

public static final String COLUMN\_AGE = "age";

Добавляем новый контакт

public void addContact(Contact contact) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_CAT\_NAME, contact.getName());

values.put(COLUMN\_PHONE, contact.getPhoneNumber());

values.put(COLUMN\_AGE, contact.getAge());

// Вставляем строку в таблицу

db.insert(DATABASE\_TABLE, null, values);

db.close();

}

Получить контакт

// Получить контакт по идентификатору

public Contact getContact(int id) {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.query(DATABASE\_TABLE, new String[] { COLUMN\_ID,

COLUMN\_CAT\_NAME, COLUMN\_PHONE, COLUMN\_AGE }, COLUMN\_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), Integer.parseInt(cursor.getString(3)));

cursor.close();

return contact;

}

Получить все контакты

public List<Contact> getAllContacts() {

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

// Выбираем всю таблицу

String selectQuery = "SELECT \* FROM " + DATABASE\_TABLE;

SQLiteDatabase db = this.getWritableDatabase();

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

// Проходим по всем строкам и добавляем в список

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));

contact.setAge(Integer.parseInt(cursor.getString(3)));

contactList.add(contact);

} while (cursor.moveToNext());

}

cursor.close();

return contactList;

}

Получить число контактов

// Получить число контактов

public int getContactsCount() {

String countQuery = "SELECT \* FROM " + DATABASE\_TABLE;

SQLiteDatabase db = this.getReadableDatabase();

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

int count = cursor.getCount();

cursor.close();

return count;

}

Обновить контакт

public int updateContact(Contact contact) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_CAT\_NAME, contact.getName());

values.put(COLUMN\_PHONE, contact.getPhoneNumber());

values.put(COLUMN\_AGE, contact.getAge());

// обновляем строку

return db.update(DATABASE\_TABLE, values, COLUMN\_ID + " = ?",

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

}

Удалить контакт

public void deleteContact(Contact contact) {

SQLiteDatabase db = this.getWritableDatabase();

db.delete(DATABASE\_TABLE, COLUMN\_ID + " = ?",

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

db.close();

}

Полностью код класса станет таким.

Показать код (щелкните мышкой)

Переходим в главную активность. При запуске приложения добавим четыре новые записи в базу. А при нажатии на кнопку получим все контакты. Будьте внимательны, при каждом запуске приложения записи заново будет вставляться в базу и она начнёт разбухать. Вы можете удалить приложение из эмулятора и запускать процесс снова. Или переместить код вставки данных в другое место.

// Если этот код работает, его написал Александр Климов,

// а если нет, то не знаю, кто его писал.

package ru.alexanderklimov.database;

import android.os.Bundle;

import android.support.v7.app.ActionBarActivity;

import android.view.View;

import android.widget.TextView;

import java.util.List;

public class MainActivity extends ActionBarActivity {

private DatabaseHelper mDatabaseHelper;

private TextView mInfoTextView;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mInfoTextView = (TextView) findViewById(R.id.textView);

mDatabaseHelper = new DatabaseHelper(this);

// Вставляем контакты

mDatabaseHelper.addContact(new Contact("Рыжик", "9100000000", 8));

mDatabaseHelper.addContact(new Contact("Васька", "9159999999", 9));

mDatabaseHelper.addContact(new Contact("Мурзик", "9022222222", 11));

mDatabaseHelper.addContact(new Contact("Барсик", "9433333333", 6));

}

public void onClick(View view) {

// Считываем все контакты

StringBuilder stringBuilder = new StringBuilder();

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

for (Contact contact : contacts) {

String allContacts = "ID: " + contact.getID() + ", Имя: "

+ contact.getName() + ", Номер: " + contact.getPhoneNumber()

+ ", Возраст: " + contact.getAge() + "\n";

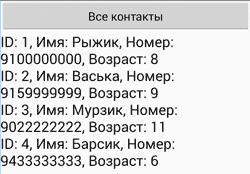
stringBuilder.append(allContacts);

}

mInfoTextView.setText(stringBuilder.toString());

}

}



Часто возвращаемые данные удобнее размещать в списках. Добавим на экран **ListView** и попробуем воспользоваться адаптером **ArrayAdapter**.

public void onClick(View view) {

// Считываем все контакты

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

ArrayAdapter<Contact> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1,

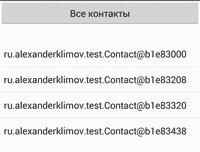
android.R.id.text1, contacts);

ListView catsListView = (ListView) findViewById(R.id.listViewCats);

catsListView.setAdapter(adapter);

}

В результате получим фигню какую-то.



Проблема заключается в том, что **ArrayAdapter<T>** работает со строками. Хотя в документации и описывается любой тип **<T>**, но в реальности из объекта берётся подходящий тип и вставляется в адаптер. Если бы в нашем объекте **Contact** не было объектов типа **String**, то приложение завершилось бы с ошибкой.

Чтобы решить эту проблему с минимальными затратами, можно переопределить метод **toString()**. Возвращаемся в класс **Contact** и добавляем новый метод:

@Override

public String toString() {

return this.mName + " (" + this.mPhoneNumber + ")";

}

Метод возвращает имя кота и его номер телефона. Идентификатор и возраст я решил не использовать. Запустим приложение ещё раз.

