**Подключение к базе данных SQLite**

В Android имеется встроенная поддержка одной из распространенных систем управления базами данных - SQLite. Для этого в пакете **android.database.sqlite** определен набор классов, которые позволяют работать с базами данных SQLite. И каждое приложение может создать свою базу данных.

Чтобы использовать SQLite в Android, надо создать базу данных с помощью выражение на языке SQL. После этого база данных будет храниться в каталоге приложения по пути:

|  |  |
| --- | --- |
| 1 | DATA/data/[Название\_приложения]/databases/[Название\_файла\_базы\_данных] |

ОС Android по умолчанию уже содержит ряд встроенных бад SQLite, которые используются стандартными программами - для списка контактов, для хранения фотографий с камеры, музыкальных альбомов и т.д.

Основную функциональность по работе с базами данных предоставляет пакет **android.database**. Функциональность непосредственно для работы с SQLite находится в пакете **android.database.sqlite**.

База данных в SQLite представлена классом **android.database.sqlite.SQLiteDatabase**. Он позволяет выполнять запросы к бд, выполнять с ней различные манипуляции.

Класс **android.database.sqlite.SQLiteCursor** предоставляет запрос и позволяет возвращать набор строк, которые соответствуют этому запросу.

Класс **android.database.sqlite.SQLiteQueryBuilder** позволяет создавать SQL-запросы.

Сами sql-выражения представлены классом **android.database.sqlite.SQLiteStatement**, которые позволяют с помощью плейсхолдеров вставлять в выражения динамические данные.

Класс **android.database.sqlite.SQLiteOpenHelper** позволяет создать базу данных со всеми таблицами, если их еще не существует.

В SQLite применяется следующая система типов данных:

* **INTEGER**: представляет целое число, аналог типу int в java
* **REAL**: представляет число с плавающей точкой, аналог float и double в java
* **TEXT**: представляет набор символов, аналог String и char в java
* **BLOB**: представляет массив бинарных данных, например, изображение, аналог типу int в java

Сохраняемые данные должны представлять соответствующие типы в java.

**Создание и открытие базы данных**

Для создания или открытия новой базы данных из кода Activity в Android мы можем вызвать метод **openOrCreateDatabase()**. Этот метод может принимать три параметра:

* название для базы данных
* числовое значение, которое определяет режим работы (как правило, в виде константы MODE\_PRIVATE)
* необязательный параметр в виде объекта SQLiteDatabase.CursorFactory, который представляет фабрику создания курсора для работы с бд

Например, создание базы данных app.db:

|  |  |
| --- | --- |
| 1 | SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE\_PRIVATE, null); |

Для выполнения запроса к базе данных можно использовать метод **execSQL** класса SQLiteDatabase. В этот метод передается SQL-выражение. Например, создание в базе данных таблицы users:

|  |  |
| --- | --- |
| 1  2 | SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE\_PRIVATE, null);  db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age INTEGER)"); |

Если нам надо не просто выполнить выражение, но и получить из бд какие-либо данные, то используется метод **rawQuery()**. Этот метод в качестве параметра принимает SQL-выражение, а также набор значений для выражения sql. Например, получение всех объектов из базы данных:

|  |
| --- |
| SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE\_PRIVATE, null);  db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age INTEGER)");  Cursor query = db.rawQuery("SELECT \* FROM users;", null);  if(query.moveToFirst()){        String name = query.getString(0);      int age = query.getInt(1);  } |

Метод db.rawQuery() возвращает объект Cursor, с помощью которого мы можем извлечь полученные данные.

Возможна ситуация, когда в базе данных не будет объектов, и для этого методом query.moveToFirst() пытаемся переместиться к первому объекту, полученному из бд. Если этот метод возвратит значение false, значит запрос не получил никаких данных из бд.

Теперь для работы с базой данных сделаем простейшее приложение. Для этого создадим новый проект.

В файле **activity\_main.xml** определим простейший графический интерфейс:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"      android:id="@+id/activity\_main"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      android:padding="16dp"      android:orientation="vertical">      <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Click"          android:onClick="onClick"/>      <TextView          android:id="@+id/textView"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:textSize="18sp" />  </LinearLayout> |

А в классе **MainActivity** определим взаимодействие с базой данных:

|  |
| --- |
| package com.example.eugene.sqliteapp;    import android.database.Cursor;  import android.database.sqlite.SQLiteDatabase;  import android.support.v7.app.AppCompatActivity;  import android.os.Bundle;  import android.view.View;  import android.widget.TextView;    public class MainActivity extends AppCompatActivity {        @Override      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);      }        public void onClick(View view){          SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE\_PRIVATE, null);          db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age INTEGER)");          db.execSQL("INSERT INTO users VALUES ('Tom Smith', 23);");          db.execSQL("INSERT INTO users VALUES ('John Dow', 31);");            Cursor query = db.rawQuery("SELECT \* FROM users;", null);          TextView textView = (TextView) findViewById(R.id.textView);          if(query.moveToFirst()){              do{                  String name = query.getString(0);                  int age = query.getInt(1);                  textView.append("Name: " + name + " Age: " + age + "\n");              }              while(query.moveToNext());          }          query.close();          db.close();      }  } |

По нажатию на кнопку здесь вначале создается в базе данных app.db новая таблица users, а затем в нее добавляются два объекта в базу данных с помощью SQL-выражения INSERT.

Далее с помощью выражения SELECT получаем всех добавленных пользователей из базы данных в виде курсора Cursor.

Вызовом query.moveToFirst() перемещаемся в курсоре к первому объекту, и так как у нас может быть больше одного объекта, то проходим по всему курсору в цикле do...while.

Для получения данных из курсора применяются методы query.getString(0) и query.getInt(1). В скобках в методы передается номер столбца, из которого мы получаем данные. Например, выше мы добавили вначале имя пользователя в виде строки, а затем возраст в виде числа. Значит, нулевым столбцом будет идти строкое значение, которое получаем с помощью метода getString(), а следующим - первым столбцом идет числовое значение, для которого применяется метод getInt().

После завершения работы с курсором и базой данных мы закрываем все связанные объекты:

|  |  |
| --- | --- |
| 1  2 | query.close();  db.close(); |

Если мы не закроем курсор, то можем столкнуться с проблемой утечки памяти.

И если мы обратимся к приложению, то после нажатия на кнопку в текстовое поле будут выведены добавленные данные:

