Курс:
Программирование под Android с нуля

Тренер:
Пономарев Алексей
oleksii_android@web-academy.com.ua

WEB ACADEMY
PROGRAMMING COURSES

Лекция 10
SQLite

Реляционные базы данных

- **База данных (БД)** – набор взаимосвязанных таблиц. Каждая колонка в таблице имеет свой тип.
- **СУБД** – система управления базами данных.
- **Запись** – строка таблицы. Как правило, одна строка это одна сущность (один студент, один автомобиль и т.д.)
- **Первичный ключ** – уникальный идентификатор строки

Типы данных

- **INTEGER** – целое число
- **REAL** – дробное число
- **TEXT** – текст
- **BLOB** – двоичные данные

SQL Online

http://www.w3schools.com/sql

Создание таблиц

```
CREATE TABLE Students
(
      _id            INTEGER    PRIMARY KEY  AUTOINCREMENT,
      FirstName   TEXT         NOT NULL,
      LastName   TEXT         NOT NULL,
      Age           INTEGER   NOT NULL
);
```

- **CREATE TABLE Students** – создать таблицу Students
- **_id** – название колонки
- **INTEGER** – тип колонки
- **PRIMARY KEY** – колонка является первичным ключом
- **AUTOINCREMENT** – значение колонки автоинкрементное
- **NOT NULL** – колонка не может быть пустой

Вставка

```
INSERT INTO Students(FirstName, LastName, Age)
VALUES
('Ivan', 'Ivanov', 22),
('Petro', 'Petrov', 23)
```

- **INSERT INTO** – вставка
- **Students** – таблица
- **FirstName, LastName, Age** – колонки таблицы

Чтение

```
SELECT * FROM Students   //Какие колонки и откуда
WHERE Age > 20           //Условие
ORDER BY FirstName       //Сортировка
```

```
SELECT FirstName, LastName FROM Students
WHERE Age > 20 AND LastName != "Ann"
ORDER BY FirstName
```

```
SELECT Age, FirstName, LastName FROM Students
WHERE Age > 20 OR Age < 20
ORDER BY FirstName
```

Обновление

```
UPDATE Students
SET Age = 20;
```

```
UPDATE Students
SET Age = 20, FirstName = 'Alex';
```

```
UPDATE Students
SET Age = 20
WHERE Age < 20
```

Удаление

```
//Удалить все из таблицы
DELETE FROM Students
```

```
//Удалить не все
DELETE FROM Students
WHERE id = 1
```

```
//Удалить таблицу
DROP TABLE Students
```

Связи

```
CREATE TABLE Groups
(
      _id              INTEGER  PRIMARY KEY AUTOINCREMENT,
      Number       TEXT             NOT NULL
);
```

```
CREATE TABLE Students
(
       _id              INTEGER    PRIMARY KEY   AUTOINCREMENT,
       _idGroup     INTEGER          NOT NULL,
      FirstName    TEXT               NOT NULL,
      LastName     TEXT               NOT NULL,
      Age              INTEGER          NOT NULL,
      FOREIGN KEY (_idGroup) REFERENCES Groups(_id)
     //Связь между таблицами
);
```

Объединения

```
SELECT * FROM Groups g
INNER JOIN Students s on g._id = s._idGroup;
```

```
SELECT g.Number, s.FirstName, s.LastName FROM Groups g
INNER JOIN Students s on g._id = s._idGroup
WHERE g.Number LIKE '%1%';
```

```
SELECT g.Number, s.FirstName, s.LastName FROM Groups g
LEFT JOIN Students s on g._id = s._idGroup;
```

- **INNER JOIN** – все записи из первой таблицы, для которых есть записи во второй таблице
- **LEFT JOIN** – все записи из первой таблицы, и все что можно из второй

SQLiteOpenHelper

```
public class DatabaseHelper extends SQLiteOpenHelper {
    public DatabaseHelper(Context context) {
       //Создание базы данных
        super(context, "MyDB.db", null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
       //Создание таблиц
        db.execSQL("create table Groups ("
                + "_id integer primary key autoincrement,"
                + "Number integer not null);");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db,
       int oldVersion, int newVersion) {
       //Обновление базы данных
    }
}
```

Вставка

```
//Новая группа
ContentValues cvNewGroup = new ContentValues();
cvNewGroup.put("Number", "1");

//Вставка группы
long groupId = db.insert("Groups", null, cvNewGroup);
```

```
//Новый студент
ContentValues cvNewStudent = new ContentValues();
cvNewStudent.put("idGroup", groupId);
cvNewStudent.put("FirstName", "Ivan");
cvNewStudent.put("LastName", "Ivanov");
cvNewStudent.put("Age", 22);

//Вставка студента
db.insert("Students", null, cvNewStudent);
```

Обновление

```
ContentValues cvUpdStudent = new ContentValues();
cvUpdStudent.put("FirstName", "Petro");
cvUpdStudent.put("LastName", "Petrov");
cvUpdStudent.put("Age", 21);

db.update("Students", cvUpdStudent, "_id=1", null);
```

или

```
db.update("Students", cvUpdStudent,
      "_id=? and FirstName=?",
      new String[]{"1", "'Ivan'" });
```

или

```
db.update("Students", cvUpdStudent,
      "_id=1 and FirstName='Ivan'", null);
```

Прочитать всех

```
Cursor studentsCursor = db.query("Students", null, null, null,
        null, null, null);

studentsCursor.moveToFirst();
while (!studentsCursor.isAfterLast()) {
 Student student = new Student();

 student.id =
   studentsCursor.getLong(studentsCursor.getColumnIndex("_id"));
 student.idGroup =
   studentsCursor.getLong(studentsCursor.getColumnIndex("idGroup"));
 student.FirstName =
   studentsCursor.getString(studentsCursor.getColumnIndex("FirstName"));
 student.LastName =
   studentsCursor.getString(studentsCursor.getColumnIndex("LastName"));
 student.Age =
   studentsCursor.getLong(studentsCursor.getColumnIndex("Age"));

 students.add(student);
 studentsCursor.moveToNext();
}

studentsCursor.close();
```

Прочитать одного

```
Student student = new Student();
Cursor studentCursor = db.query("Students",
        new String[]{"_id", "idGroup", "Name", "Age"},
        "_id=1",
        null, null, null, null);

if (studentCursor.moveToFirst()) {
  student.id =
    studentCursor.getLong(studentCursor.getColumnIndex("_id"));
  student.idGroup =
   studentCursor.getLong(studentCursor.getColumnIndex("idGroup"));
  student.FirstName =
    studentCursor.getString(studentCursor.getColumnIndex("FirstName"));
  student.LastName =
    studentCursor.getString(studentCursor.getColumnIndex("LastName"));
  student.Age =
    studentCursor.getLong(studentCursor.getColumnIndex("Age"));
}

studentCursor.close();
```

JOIN

```
ArrayList<Student> students = new ArrayList<Student>();

String tables = "Groups as g inner join Students as s on g._id = s.idGroup";
String columns[] = { "s._id", "s.FirstName", "s.LastName", "s.Age" };
String where = "g.Number = '1'";
String orderBy = "s.FirstName";

Cursor joinCursor = db.query(tables, columns, where,
        null, null, null, orderBy, null);

joinCursor.moveToFirst();
while (!joinCursor.isAfterLast()) {
    Student student = new Student();

    student.id = joinCursor.getLong(joinCursor.getColumnIndex("_id"));
    student.FirstName =
joinCursor.getString(joinCursor.getColumnIndex("FirstName"));
    student.LastName =
joinCursor.getString(joinCursor.getColumnIndex("LastName"));
    student.Age = joinCursor.getLong(joinCursor.getColumnIndex("Age"));

    students.add(student);
   joinCursor.moveToNext();
}

joinCursor.close();
```

Удалить

```
db.delete("Students", "_id=1", null);

db.delete("Students", null, null);
```

SimpleCursorAdapter

```
ListView listView = (ListView) findViewById(R.id.listView);

DatabaseHelper DBHelper = new DatabaseHelper(this);
SQLiteDatabase db = DBHelper.getWritableDatabase();

Cursor studentsCursor = db.query("Students",
       null, null, null, null, null, null);

SimpleCursorAdapter adapter = new SimpleCursorAdapter(
        this,
        android.R.layout.simple_list_item_2,
        studentsCursor,
        new String[]{"FirstName", "LastName"},
        new int[]{android.R.id.text1, android.R.id.text2});

listView.setAdapter(adapter);
```

Спасибо за внимание

Тренер:
Пономарев Алексей
oleksii_android@web-academy.com.ua

WEB ACADEMY
PROGRAMMING COURSES

Лекция 10
SQLite