МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»



**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

по выполнению лабораторных работ

по дисциплине «Программирование мобильных устройств»

для студентов направлений 09.03.03 «Прикладная информатика» и 09.03.02 «Информационные системы и технологии».

Ставрополь

2021

# **Лабораторная работа № 4. Использование базы данных.**

**Цель работы:** Научиться использовать встроенную базу данных

**Формируемые компетенции:** ПК-7, ПК-8

**Теоретическая часть**

В наше время практически не существует приложения, которое бы не использовало базы данных. Магазины, конструкторы, игры, всё упирается в их использование. В мобильных устройствах также существует способ хранить данные. СУБД (Система управления базами данных), используемая в телефонах называет SQLite. SQLite –упрощённая версия реляционной СУБД, обладающая базовыми возможностями полноценной СУБД.

В SQLite используются следующие типы данных:

* Null
* Integer
* Real
* Text
* Blob(бинарные)
* Numeric.

Несложно заметить, что SQLite не работает с логическими переменными Однако, сложно в наше время найти базу, которая не содержала бы значения true или false. Чтобы обойти данную "оплошность", такие значения следует хранить как обычный integer, где 1 выполняет роль true и 0 – false.

Для даты есть два способа хранения:

* Строка – ‘2013-03-27T07:58’ (Здесь используется стандартный формат даты, используемые в Америке, то есть год-месяц-день);
* Число – количество секунд с 1970-01-01Т00:00:00.

Когда ваше приложение создаёт базу данных, она сохраняется в каталоге DATA/data/имя\_пакета/databases/имя\_базы.db.

Environment.getDataDirectory() возвращает путь к каталогу DATA.

Ход работы:

1. Для работы с SQLite необходимо создать «посредника», который бы делал всю работу сам, а мы бы просто пользовались его услугами. Таким посредником будет являться класс SQLHelper. Чтобы его создать, переходим к директории app/java и создадим в той же папке, где находится MainActivity, новый класс.

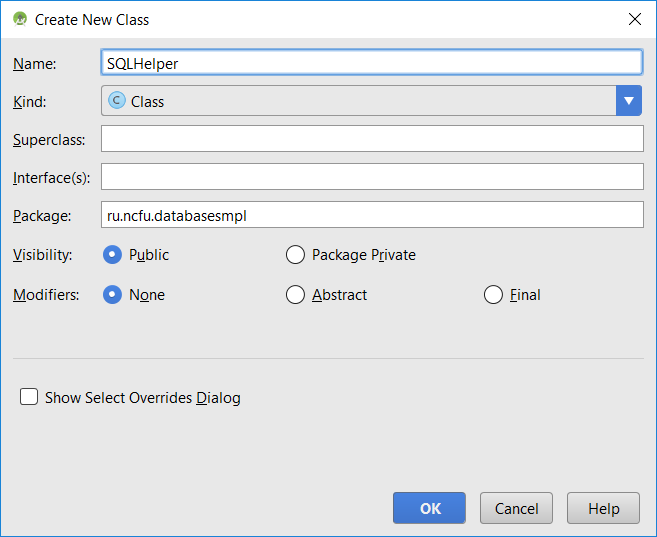


Рисунок 1 – Создание класса

SQLHelper должен наследоваться от SQLiteOpenHelper.

1. Создадим сразу же конструктор класса.

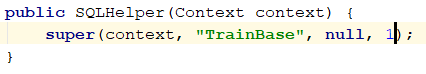


Рисунок 2 – Конструктор

Передаём в него контекст нашей страницы, название базы данных, фабрику и версию. Фабрика (CursorFactory) создаёт особые курсоры для БД. Вам это не пригодится. Версия – показывает текущую версию Вашей БД. Вначале версия будет равна 1. Если же, например, какие-то таблицы меняли свою структуру, то стоит повышать её версию на 1 (после первого изменения будет равно 2).

1. Переопределим методы onCreate() и onUpgrade().

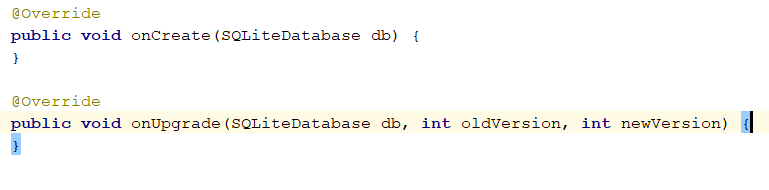


Рисунок 3 – Перегрузка методов

Во время создания БД будет вызван метод onCreate(). В него следует поместить запрос на создание первоначальных таблиц.

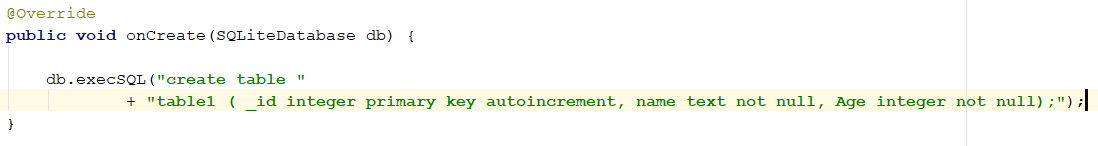


Рисунок 4 – Создание БД

В случае изменения структуры, будет вызван соответственно метод onUpgrade(). Он принимает 3 параметра: новую базу данных, номер старой версии и номер новой версии. Соответственно, можно использовать это для создания некоторых ограничений. Однако, в данном примере нам ничего не нужно менять.

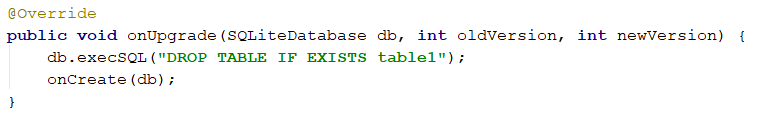


Рисунок 5 – Обновление БД

База данных не создаётся конкретно в этот момент, поэтому выполняется этот метод быстро. Непосредственное создание/открытие БД происходит после вызова одного из двух методов:

* getReadableDatabase() – этот метод проверяет, возможно ли получить доступ к базе данных, с целью прочесть данные. Если можно, то он вернёт базу, иначе, выдаст исключение.
* getWritableDatabase() – этот метод проверяет, возможно ли получить доступ к базе данных, с целью не только прочесть, но и записать данные в БД. Также, возвращает саму базу, если успешно.

Так как с БД порой случаются ошибки, связанные с нехваткой памяти или прав доступа, getWritableDatabase() может вызвать ошибку, которая закроет приложение. Чтобы предотвратить это, следует просто обернуть конструкцию в try/catch, вызвать getReadableDatabase() и сообщить пользователю, что не удалось подключиться для записи.

Для чтения/изменения данных обычно используются следующие методы:

* query(), rawQuery() – запрос на выборку данных;
* insert() – добавить данные в таблицу;
* delete() – удалить данные из таблицы;
* update() –обновить данные в таблице;
* execSQL(запрос) – запрос на изменение структуры таблицы.

Метод query() принимает 7 параметров:

* название таблицы;
* массив атрибутов (колонки);
* ограничение where (вместо фиксированных значений можно ставить “?”);
* массив динамических данных, используемых в where;
* groupBy;
* having;
* orderBy.

Если какой-либо из параметров Вам не нужен, необходимо добавить null.

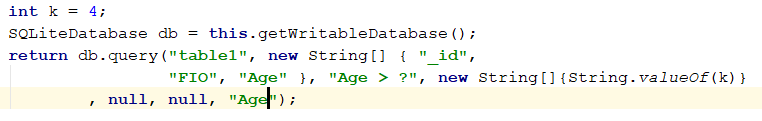


Рисунок 6 – Пример запроса

Агрегатные функции также могут быть использованы в запросе.

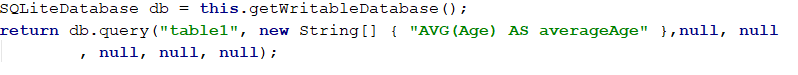


Рисунок 7 – Пример использования агрегатных функций

Альтернативой для query() является метод rawQuery(). Для него не нужно разбивать всё по параметрам, так как он принимает лишь строку запроса, как в обычной БД.



Рисунок 8 – использование rawQuery

Метод insert() принимает 3 параметра:

* название таблицы;
* Особый параметр для столбцов, которым будет присвоено значение null. Если необходимо вставить пустую строку в базу данных, то следует указать название хотя бы одного атрибута. Если строка не пустая, можно оставить это значение null.
* Объект типа ContentValues – собственно, пары значений название атрибута - значение атрибута.

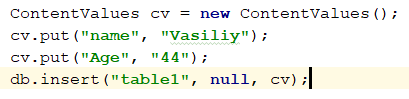


Рисунок 9 – Пример insert

Метод update() принимает 4 параметра:

* Название таблицы;
* Данные ContentValues;
* Ограничение where;
* Динамические параметры для ограничения where.

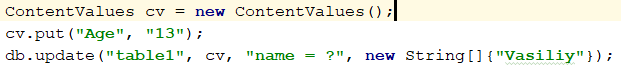
****

Рисунок 10 – Использование where

Последний метод, delete() принимает 3 параметра:

* Название таблицы;
* Ограничение where;
* Динамические параметры для ограничения where.

1. Возвращаемся в класс MainActivity. Создадим экземпляр посредника БД.

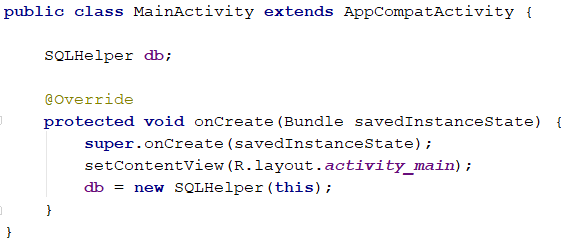


Рисунок 11 –Создание объекта SQLHelper

1. В целях соблюдения логики, для каждой таблицы следует создавать свой класс (Модель), чтобы описать в нём атрибуты объектов. Создадим новый класс в той же папке с названием Person.

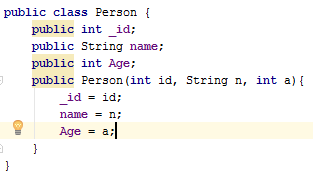


Рисунок 12 – Создание модели

1. Создадим метод, чтобы получить текущий список объектов в БД.

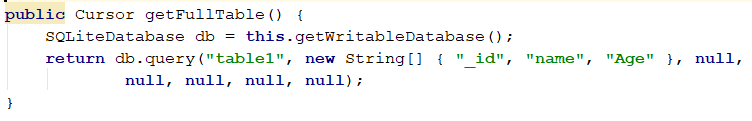


Рисунок 13 – Получение всех объектов

1. Теперь обработаем полученный список и выведем его на дисплее телефона.

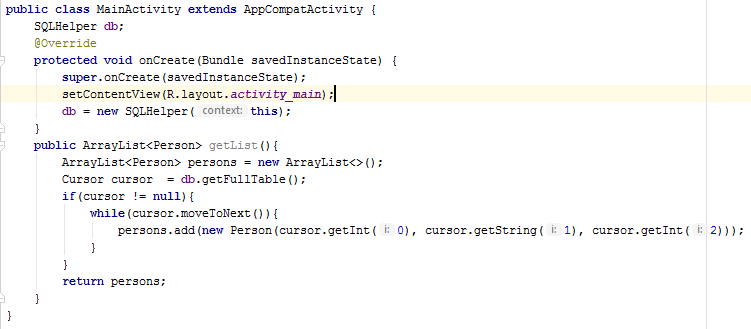


Рисунок 14 – Вывод на экран

ArrayList - динамический массив данных в языке java.

Методы getInt() и getString() автоматически преобразовывают форматы с БД в форматы для java. В скобках передаётся идентификатор атрибута, т.е., если мы выбираем Id и Name в таком порядке, то под 0 можно будет получить Id, а под 1 – имя. Иногда бывают случаи, когда неизвестно, что придёт под каким номером (например, если идёт выборка всех атрибутов). Для решения этой задачи используется метод getColumnIndex().

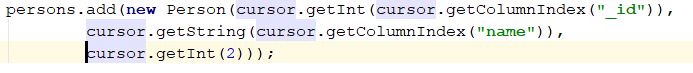


Рисунок 15 – Пример использование getColumnIndex

В файле xml создадим LinearLayout, который будет отображать наш список людей.

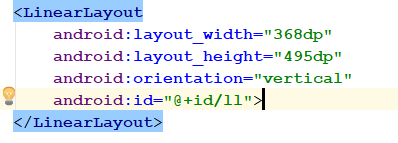


Рисунок 16 – Создание менеджера для отображения данных

В методе onCreate() класса MainActivity вызовем созданный метод и результат поместим в LinearLayout.



Рисунок 17 – Обработка полученных данных с базы

В результате, получится следующее.

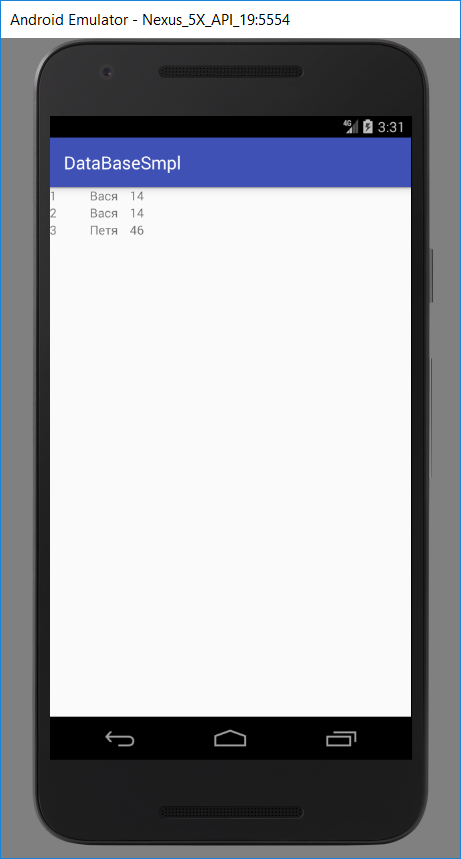


Рисунок 18 – Результат выполнения

**Оборудование и материалы:** для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 (8,8.1,10) и программным продуктом Android Studio.

**Указания по технике безопасности:** к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

**Задания:**для выполнения лабораторной работы необходимо выполнить следующее:

1. Реализовать структуру БД согласно варианту.
2. Создать методы для отображения и редактирования некоторых данных, взятых из БД.

**Варианты**

1. Приёмная комиссия университета.
2. Школа.
3. База таксистов.
4. Центр занятости.
5. Поликлиника.
6. Магазин электроники.
7. Магазин книг.
8. ЖКХ.
9. Железнодорожная станция.
10. Ветеринарная клиника.
11. Швейная компания.
12. Магазин продажи авто.
13. Служба ремонта телефонов.

**Содержание отчета:** отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выровнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

**Контрольные вопросы:**

1. Типы данных SQLite
2. Методы работы с SQLite
3. ООП
4. ContentValues

**Список литературы, рекомендуемый к использованию по данной теме:**

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book\_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book\_red&id=442808&sr=1