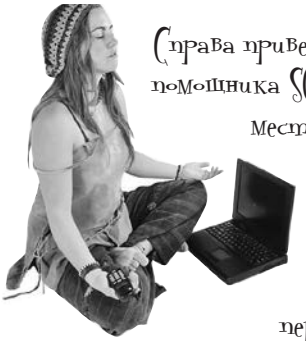
СТАНЬ помощником SQLite



Справа приведены примеры кода помощника SQLite. Представьте себя на месте помощника SQLite и скажите, какой из сегментов кода (из помеченных буквами) будет выполняться для каждого из пользователей, перечисленных ниже. Мы решили одну задачу за Вас, чтобы Вам было проще взяться за дело.

```
...
class MyHelper extends SQLiteOpenHelper{

    StarbuzzDatabaseHelper(Context context){
        super(context, "fred", null, 4);
    }

    @Override
    public void onCreate(SQLiteDatabase db){
      A //Выполняется код A
        ...
    }

    @Override
    public void onUpgrade(SQLiteDatabase db,
                          int oldVersion,
                          int newVersion){
        if (oldVersion < 2) {
          B //Выполняется код B
            ...
        }
        if (oldVersion == 3) {
          C //Выполняется код C
            ...
        }
      D //Выполняется код D
        ...
    }

    @Override
    public void onDowngrade(SQLiteDatabase db,
                            int oldVersion,
                            int newVersion){
        if (oldVersion == 3) {
          E //Выполняется код E
            ...
        }
        if (oldVersion < 6) {
          F //Выполняется код F
            ...
        }
    }
}
```

Пользователь 1 запускает приложение впервые.

Сегмент A. На устройстве пользователя нет базы данных, выполняется метод onCreate().

У пользователя 2 установлена база данных с номером версии 1.

У пользователя 3 установлена база данных с номером версии 2.

У пользователя 4 установлена база данных с номером версии 3.

У пользователя 5 установлена база данных с номером версии 4.

У пользователя 6 установлена база данных с номером версии 5.

Ответы на с. 692.

Обновление существующей базы данных

При обновлении базы данных обычно выполняются два вида действий:

1. **Изменение записей базы данных.**
Ранее в этой главе было показано, как выполнять вставку записей в базу данных методом SQLiteDatabase insert(). Обновление базы данных может сопровождаться добавлением новых записей, изменением или удалением уже существующих записей.

2. **Изменение структуры базы данных.**
Вы уже знаете, как создавать таблицы в базе данных. Также возможны операции добавления столбцов в существующие таблицы, переименования и даже полного удаления таблиц.

Начнем с изменения записей базы данных.

Как обновляются базы данных

Обновление записей в таблице выполняется по тому же принципу, что и вставка.

Все начинается с создания объекта `ContentValues`, определяющего новые значения обновляемых полей. Например, предположим, что данные Latte в таблице DRINK обновляются таким образом, чтобы поле DESCRIPTION содержало значение «Tasty»:

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	"Latte"	"Espresso and steamed milk" "Tasty"	54543543

Для этого следует создать новый объект `ContentValues` с описанием обновляемых данных:

```
ContentValues drinkValues = new ContentValues();
drinkValues.put("DESCRIPTION", "Tasty");
```

Значение столбца DESCRIPTION заменяется строкой Tasty, поэтому имя «DESCRIPTION» связывается со значением «Tasty».

При обновлении записей в объекте `ContentValues` достаточно указать только данные, которые должны изменяться, а не всю строку данных.

После добавления изменяемых данных в объект `ContentValues` метод `update()` класса `SQLiteDatabase` используется для обновления данных. Этот метод описан на следующей страниц.

Обновление записей методом update()

Для обновления существующей информации в SQLite используется метод `update()`. Этот метод вносит изменения в записи, хранящиеся в базе данных, и возвращает количество обновленных записей. Чтобы использовать метод `update()`, необходимо указать таблицу, в которой обновляются значения, объект `ContentValues` с обновляемыми значениями и условия их обновления.

В следующем примере значение столбца DESCRIPTION заменяется строкой «Tasty», если поле названия напитка содержит «Latte»:

```
ContentValues drinkValues = new ContentValues();
drinkValues.put("DESCRIPTION", "Tasty");
db.update("DRINK",
          drinkValues,
          "NAME = ?",
          new String[] {"Latte"});
```

Имя таблицы, в которой обновляются записи.

Объект ContentValues с новыми значениями.

Условия обновления данных (в данном случае NAME = «Latte»).

Значение «Latte» подставляется вместо ? в конструкции «NAME = ?».

Первый параметр метода `update()` содержит имя таблицы, в которой обновляется информация (в данном случае таблица DRINK).

Во втором параметре передается объект `ContentValues` с описанием обновляемых значений. В приведенном выше примере в объект `ContentValues` были добавлены значения `"DESCRIPTION"` и `"Tasty"`, поэтому значение столбца DESCRIPTION будет заменено строкой «Tasty».

Последние два параметра определяют, в каких записях должно происходить обновление; для этого они описывают условие выбора записей. В сочетании они образуют секцию `WHERE` команды SQL.

Третий параметр задает имя столбца в условии. В приведенном примере должны обновляться записи, у которых столбец NAME содержит значение «Latte», поэтому используется запись `"NAME = ?"`; это означает, что значение столбца NAME должно быть равно некоторому значению. Новое значение подставляется на место знака «?».

В последнем параметре передается массив строк со значениями условий. В нашем примере обновляются записи, у которых столбец NAME содержит строку «Latte»:

```
new String[] {"Latte"});
```

Более сложные условия рассматриваются на следующей странице.

Будьте осторожны!

Если в двух последних параметрах **update()** *передается значение* **null**, *будут обновлены ВСЕ записи в таблице.*

Например, вызов:

```
db.update("DRINK",
          drinkValues,
          null, null);
```

обновит все данные в таблице DRINK.

Определение условий по нескольким столбцам

Также можно определять условия, применимые к нескольким столбцам. Например, вот как выполняется обновление записей из таблицы DRINK, у которых столбец названия напитка содержит текст «Latte» или столбец описания содержит текст «Our best drip coffee».

```
db.update("DRINK",
          drinkValues,
          "NAME = ? OR DESCRIPTION = ?",
          new String[] {"Latte", "Our best drip coffee"});
```

Это означает: Если NAME = «Latte» или DESCRIPTION = «Our best drip coffee».

Каждый знак «?» заменяется значением из массива. Количество значений в массиве должно совпадать с количеством знаков «?».

Если вы хотите определить условия, распространяющиеся на несколько столбцов, имена этих столбцов следует передать в третьем параметре метода `update()`. Как и прежде, вместо значений, включаемых в условие, добавляются знаки «?». После этого фактические значения указываются в четвертом параметре метода `update()`.

Значения условий должны относиться к строковому типу String, даже если столбец, к которому относится условие, содержит данные другого типа. В таких случаях значения необходимо преобразовать к типу String. Например, следующий вызов возвращает записи DRINK, в которых столбец _id равен 1:

```
db.update("DRINK",
          drinkValues,
          "_id = ?",
          new String[] {Integer.toString(1)});
```

Целое значение 1 преобразуется в String.

Удаление записей методом delete()

Для удаления записей используется метод `delete()` класса `SQLiteDatabase`. Он работает по тому же принципу, что и только что рассмотренный метод `update()`: вы указываете таблицу, из которой удаляются записи, и условие удаления. Например, следующая команда удаляет из таблицы DRINK все записи, у которых столбец названия содержит текст «Latte»:

```
db.delete("DRINK",
          "NAME = ?",
          new String[] {"Latte"});
```

Видите, как это похоже на метод update()?

Удаляется вся строка данных.

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	"Latte"	"Espresso and steamed milk"	54543543

В первом параметре передается имя таблицы, из которой удаляются записи (в данном случае DRINK). Два других параметра позволяют точно описать, какие записи требуется удалить (NAME = «Latte»).

упражнение

Возьми в руку карандаш

Ниже приведен метод `onCreate()` класса `SQLiteOpenHelper`. Укажите, какие значения будут вставлены в столбцы NAME и DESCRIPTION таблицы DRINK после того, как метод onCreate() завершит свою работу.

```
@Override
public void onCreate(SQLiteDatabase db) {
    ContentValues espresso = new ContentValues();
    espresso.put("NAME", "Espresso");
    ContentValues americano = new ContentValues();
    americano.put("NAME", "Americano");
    ContentValues latte = new ContentValues();
    latte.put("NAME", "Latte");
    ContentValues filter = new ContentValues();
    filter.put("DESCRIPTION", "Filter");
    ContentValues mochachino = new ContentValues();
    mochachino.put("NAME", "Mochachino");

    db.execSQL("CREATE TABLE DRINK ("
               + "_id INTEGER PRIMARY KEY AUTOINCREMENT, "
               + "NAME TEXT, "
               + "DESCRIPTION TEXT);");
    db.insert("DRINK", null, espresso);
    db.insert("DRINK", null, americano);
    db.delete("DRINK", null, null);
    db.insert("DRINK", null, latte);
    db.update("DRINK", mochachino, "NAME = ?", new String[] {"Espresso"});
    db.insert("DRINK", null, filter);
}
```

Указывать значение столбца _id не нужно.

_id	NAME	DESCRIPTION

Ответы на с. 693.

Изменение структуры базы данных

Кроме создания, обновления и удаления записей базы данных также может возникнуть необходимость в изменении структуры базы данных. Представьте, что в нашем примере в таблицу DRINK пришлось добавить новый столбец FAVORITE.

Добавление новых столбцов средствами SQL

Ранее в этой главе было показано, как создавать таблицы командой SQL `CREATE TABLE`:

```
CREATE TABLE DRINK (_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    NAME TEXT,
                    DESCRIPTION TEXT,
                    IMAGE_RESOURCE_ID INTEGER)
```

Столбец _id является первичным ключом.

Имя таблицы.

Столбцы таблицы.

Язык SQL также может использоваться для изменения существующих таблиц — эта задача решается командой `ALTER TABLE`. Например, команда добавления столбца в таблицу выглядит так:

```
ALTER TABLE DRINK
ADD COLUMN FAVORITE NUMERIC
```

Имя таблицы.

Добавляемый столбец.

В этом примере в таблицу DRINK добавляется столбец с именем FAVORITE, в котором хранятся числовые значения.

Переименование таблиц

Команда `ALTER TABLE` также может использоваться для переименования таблиц. Например, следующая команда переименовывает таблицу DRINK в FOO:

```
ALTER TABLE DRINK
RENAME TO FOO
```

Текущее имя таблицы.

Новое имя таблицы.

На следующей странице мы покажем, как удалить таблицу из базы данных.