[Транзакции в SQLite. Небольшой FAQ по SQLite.](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/78-urok-38-tranzaktsii-v-sqlite.html" \o "Урок 38. Транзакции в SQLite. Небольшой FAQ по SQLite.)

Что такое БД-**транзакция**, думаю объяснять особо не надо. Она используется при работе с данными по принципу «**все или ничего**». Т.е., например, вам нужно вставить пачку данных. Но вставить надо так, чтобы или все вставилось или ничего не вставилось.  И если в процессе половина записей прошла, а другая нет – должна быть возможность **откатить**изменения.

Напишем простое приложение и исследуем возможности **SQLite**в этом плане.

Создадим проект:

**Project name**: P0381\_SQLiteTransaction  
**Build Target**: Android   
**Application name**: SQLiteTransaction  
**Package name**: ru.startandroid.develop.p0381sqlitetransaction  
**Create Activity**: MainActivity

Открываем**MainActivity.java** и пишем:

package ru.startandroid.develop.p0381sqlitetransaction;

import android.app.Activity;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.os.Bundle;

import android.util.Log;

public class MainActivity extends Activity {

  final String LOG\_TAG = "myLogs";

  DBHelper dbh;

  SQLiteDatabase db;

  /\*\* Called when the activity is first created. \*/

  public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    Log.d(LOG\_TAG, "--- onCreate Activity ---");

    dbh = new DBHelper(this);

    myActions();

  }

  void myActions() {

    db = dbh.getWritableDatabase();

    delete(db, "mytable");

    insert(db, "mytable", "val1");

    read(db, "mytable");

    dbh.close();

  }

  void insert(SQLiteDatabase db, String table, String value) {

    Log.d(LOG\_TAG, "Insert in table " + table + " value = " + value);

    ContentValues cv = new ContentValues();

    cv.put("val", value);

    db.insert(table, null, cv);

  }

  void read(SQLiteDatabase db, String table) {

    Log.d(LOG\_TAG, "Read table " + table);

    Cursor c = db.query(table, null, null, null, null, null, null);

    if (c != null) {

      Log.d(LOG\_TAG, "Records count = " + c.getCount());

      if (c.moveToFirst()) {

        do {

          Log.d(LOG\_TAG, c.getString(c.getColumnIndex("val")));

        } while (c.moveToNext());

      }

      c.close();

    }

  }

  void delete(SQLiteDatabase db, String table) {

    Log.d(LOG\_TAG, "Delete all from table " + table);

    db.delete(table, null, null);

  }

  // класс для работы с БД

  class DBHelper extends SQLiteOpenHelper {

    public DBHelper(Context context) {

      super(context, "myDB", null, 1);

    }

    public void onCreate(SQLiteDatabase db) {

      Log.d(LOG\_TAG, "--- onCreate database ---");

      db.execSQL("create table mytable ("

          + "id integer primary key autoincrement,"

          + "val text"

          + ");");

    }

    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }

  }

}

Разбираем код. Я создал несколько методов, где сгруппировал операции: **insert**– набор операций для вставки записи, **read**– чтение всех записей, **delete**– удаление всех записей. Класс **DBHelper**– для управления БД. Интересовать нас будет метод **myActions**. Сейчас в нем мы подключаемся к БД, очищаем таблицу mytable, вставляем строку с значением *val1*, выводим в лог все записи из таблицы и отключаемся.

Все сохраним, запустим приложение. Смотрим лог:

*--- onCreate Activity ---  
--- onCreate database ---  
Delete all from table mytable  
Insert in table mytable value = val1  
Read table mytable  
Records count = 1  
val1*

Все верно, запись вставилась и отобразилась.

Теперь попробуем использовать транзакцию. Поправим код **myActions** на этот:

void myActions() {

  db = dbh.getWritableDatabase();

  delete(db, "mytable");

  db.beginTransaction();

  insert(db, "mytable", "val1");

  db.endTransaction();

  insert(db, "mytable", "val2");

  read(db, "mytable");

  dbh.close();

}

Мы подключаемся к базе, чистим таблицу, открываем транзакцию методом [beginTransaction](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "beginTransaction()" \t "_blank), вставляем *val1*, закрываем транзакцию методом [endTransaction](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "endTransaction()" \t "_blank), вставляем *val2*, выводим содержимое в лог и отключаемся. Все сохраняем, запускаем и смотрим лог:

*--- onCreate Activity ---  
Delete all from table mytable  
Insert in table mytable value = val1  
Insert in table mytable value = val2  
Read table mytable  
Records count = 1  
val2*

По логу видно, что вставляли мы **две**записи, но прошла только **вторая**. Та, которая была в **транзакции**– не записалась. Это произошло потому, что мы явно не указали, что транзакция должна быть успешно **закрыта**. Если этого не сделать, то при закрытии транзакции все операции **отменятся**. Давайте исправимся. Снова перепишем **myActions**:

void myActions() {

  db = dbh.getWritableDatabase();

  delete(db, "mytable");

  db.beginTransaction();

  insert(db, "mytable", "val1");

  db.setTransactionSuccessful();

  insert(db, "mytable", "val2");

  db.endTransaction();

  insert(db, "mytable", "val3");

  read(db, "mytable");

  dbh.close();

}

Подключаемся к БД, чистим таблицу, открываем транзакцию, вставляем *val1*, подтверждаем успешность транзакции методом [setTransactionSuccessful](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html#setTransactionSuccessful()), вставляем *val2*, закрываем транзакцию, вставляем *val3*, выводим содержимое и отключаемся.

Сохраняем, запускаем, смотрим лог:

*--- onCreate Activity ---  
Delete all from table mytable  
Insert in table mytable value = val1  
Insert in table mytable value = val2  
Insert in table mytable value = val3  
Read table mytable  
Records count = 3  
val1  
val2  
val3*

Вставились все три записи. Обратите внимание - несмотря на то, что *val2*мы вставляли уже после подтверждения успешности транзакции, запись вставилась, вошла в эту транзакцию. Но проводить операции после подтверждения транзакции не рекомендуется хелпом.

**Транзакция**при открытии ставит **блокировку**на базу. Убедимся в этом, попробуем создать новое подключение к БД во время транзакции. Перепишем **myActions**:

void myActions() {

  try {

    db = dbh.getWritableDatabase();

    delete(db, "mytable");

    db.beginTransaction();

    insert(db, "mytable", "val1");

    Log.d(LOG\_TAG, "create DBHelper");

    DBHelper dbh2 = new DBHelper(this);

    Log.d(LOG\_TAG, "get db");

    SQLiteDatabase db2 = dbh2.getWritableDatabase();

    read(db2, "mytable");

    dbh2.close();

    db.setTransactionSuccessful();

    db.endTransaction();

    read(db, "mytable");

    dbh.close();

  } catch (Exception ex) {

    Log.d(LOG\_TAG, ex.getClass() + " error: " + ex.getMessage());

  }

}

Подключаемся к базе, чистим таблицу, открываем транзакцию, вставляем запись, создаем новое подключение к БД - **db2**, читаем содержимое вторым подключением, закрываем второе подключение, успешно закрываем транзакцию, читаем содержимое первым подключением, закрываем первое подключение.

Все сохраним и запустим. Смотрим лог:

*--- onCreate Activity ---  
Delete all from table mytable  
Insert in table mytable value = val1  
create DBHelper  
get db  
class android.database.sqlite.SQLiteException error: database is locked*

Мы видим, что при попытке создать второе подключение к базе произошла ошибка **SQLiteException**– база **заблокирована**открытой транзакцией. Если вы теперь закоментите или удалите строки управления транзакцией и снова выполните код, то все пройдет успешно, т.к. никаких блокировок не будет.

Наверняка есть некоторые вопросы по этой теме. Попробую здесь же ответить на некоторые.

close

Метод **close**есть и у **SQLiteDatabase**и у **SQLiteOpenHelper**. Какая между ними разница? Каким из них пользоваться для закрытия подключения?

Тут надо понимать один момент – объект **SQLiteOpenHelper**всегда предоставляет только **одно**подключение. Попробую объяснить этот механизм. У объекта **SQLiteOpenHelper**есть внутренний атрибут **mDatabase**типа **SQLiteDatabase**. Когда мы вызываем метод **getWritableDatabase**, объект **SQLiteOpenHelper**проверяет: если **mDatabase**не null и не закрыт, то он и идет в качестве **return**. Иначе  **SQLiteOpenHelper**выполняет подключение к БД, записывает новый **SQLiteDatabase**-объект в**mDatabase**и возвращает нам его. Т.е. метод **getWritableDatabase**либо возвращает **существующее подключение**к БД, либо создает новое в случае отсутствия подключения. Когда же выполняется метод **close**для **SQLiteOpenHelper**, то происходит вызов **close**для **mDatabase**и выполняется код **mDatabase = null**.

Рассмотрим на примере. Снова меняем метод myActions:

void myActions() {

  db = dbh.getWritableDatabase();

  SQLiteDatabase db2 = dbh.getWritableDatabase();

  Log.d(LOG\_TAG, "db = db2 - " + db.equals(db2));

  Log.d(LOG\_TAG, "db open - " + db.isOpen() + ", db2 open - " + db2.isOpen());

  db2.close();

  Log.d(LOG\_TAG, "db open - " + db.isOpen() + ", db2 open - " + db2.isOpen());

}

Сначала мы получаем **db**. При этом **dbh**проверяет свой внутренний атрибут **mDatabase**. Т.к. это первая попытка подключения, то **mDatabase**пуст, поэтому внутри **dbh**производится подключение и в **mDatabase**записывается свежесозданный**SQLiteDatabase**, и он же и возвращается в **db**из метода **getWritableDatabse**.

Затем мы из того же **dbh**получаем **db2**. **dbh**снова проверяет свой внутренний **mDatabse**, видит, что он уже **не null и не закрыт**, и возвращает нам его в наш **db2**. В итоге **db**и **db2**равны и ссылаются на один и тот же объект. Проверяем это с помощью метода **equals**. Далее проверим, что **db**и **db2 открыты**. Потом **закроем**только **db2**, и еще раз проверим на**открытость оба**объекта.

Сохраняем, запускаем, смотрим лог:

*--- onCreate Activity ---  
db = db2 – true  
db open - true, db2 open – true  
db open - false, db2 open - false*

Видим, что **equals**вернул **true**. Затем видно, что **db** и **db2 открыты**. А после закрытия **db2**видим, что закрыты оба объекта. Все оттого, что «оба объекта» – это всего лишь **две ссылки**на **один объект**.

Если в коде вместо **db2.close()** поставить**dbh.close()** - эффект будет тот же. **dbh**вызовет метод **close**для **mDatabase**и обнулит его -**mDatabase = null**. А **db**и **db2**будут ссылаться на закрытый **SQLiteDatabase**.

Я думаю, что правильнее вызывать **close**для **SQLiteOpenHelper**, а не для **SQLiteDatabase**. Т.к. гарантировано **закрывается**текущее открытое соединение и **обнуляется**внутренняя ссылка на объект.

Если вам надо получить второе открытое подключение к БД, то надо создавать новый экземпляр **DBHelper**и вызывать **getWritableDatabase**. Мы так делали чуть выше в примере с блокировкой транзакции.

read write

В чем разница между **getWritableDatabase**и **getReadableDatabase**? Судя по хелпу, в обычной ситуации **оба**метода возвращают**одно и то же**. И оба позволят **читать**и **менять**БД. В случае же, например, проблемы**отсутствия свободного места** на устройстве, метод **getReadableDatabase**вернет БД только **для чтения**, а **getWritableDatabase**выдаст **ошибку**.

\_id, как имя поля-идентификатора

В различных источниках при работе с БД в качестве наименования поля-идентификатора в таблице используют не просто id, а \_id. Почему?

Ответ нашелся в доках по Cursor-адаптерам. Цитата: "The Cursor must include a column named "\_id" or this class will not work.". Т.е. если вы планируете использовать Cursor-адаптеры, то необходимо, чтобы таблица содержала поле \_id, иначе адаптер не будет работать.

Блокировка

Метод открытия транзакции **beginTransaction**ставит блокировку в режиме **EXCLUSIVE**. Т.е. БД блокируется и на чтение и на запись для других подключений. В SDK Android версии старше 2.3.3 появился метод **beginTransactionNonExclusive**, который ставит блокировку в режиме **IMMEDIATE**. Я подозреваю, что это позволит читать данные другим подключениям.

Если есть желание подробнее погрузиться в тему, вам [сюда](http://www.enterra.ru/blog/android_issues_with_sqlite/).

Синтаксис

И кстати, рекомендуемая форма для использования транзакций такая:

db.beginTransaction();

try {

  ...

  db.setTransactionSuccessful();

} finally {

  db.endTransaction();

}

Это очень важно! Т.е. если вы **открыли**транзакцию, **выполнили**какие-либо действия и**не закрыли** транзакцию, то все операции будут считаться **неуспешными**и изменения не будут внесены в БД. Поэтому **закрытие** транзакции **необходимо** выполнять и **finally**нам это гарантирует.