Android数据库高手秘籍(3)：使用LitePal升级表

2015-09-07 安卓应用频道

(点击上方蓝字，可快速关注我们)

> 来源：郭霖
> 网址：http://blog.csdn.net/guolin_blog/article/details/39151617

《Android数据库高手秘籍(1)：SQLite命令》
《Android数据库高手秘籍(2)：创建表和LitePal的基本用法》

在上一篇文章中，我们学习了LitePal的基本用法，体验了使用框架来进行创建表操作的便利。然而大家都知道，创建表只是数据库操作中最基本的一步而已，我们在一开始创建的表结构，随着需求的变更，到了后期是极有可能需要修改的。因此，升级表的操作对于任何一个项目也是至关重要的，那么今天我们就一起来学习一下，在Android传统开发当中升级表的方式，以及使用LitePal来进行升级表操作的用法。如果你还没有看过前一篇文章，建议先去参考一下 Android数据库高手秘籍(二)——创建表和LitePal的基本用法 。

LitePal的项目地址是：https://github.com/LitePalFramework/LitePal

传统的升级表方式

上一篇文章中我们借助MySQLiteHelper已经创建好了news这张表，这也是demo.db这个数据库的第一个版本。然而，现在需求发生了变更，我们的软件除了能看新闻之外，还应该允许用户评论，所以这时就需要对数据库进行升级，添加一个comment表。

该怎么做呢？添加一个comment表的建表语句，然后在onCreate()方法中去执行它？没错，这样的话，两张表就会同时创建了，代码如下所示：

```
public class MySQLiteHelper extends SQLiteOpenHelper {

public static final String CREATE_NEWS = "create table news ("
 + "id integer primary key autoincrement, "
 + "title text, "
 + "content text, "
 + "publishdate integer,"
 + "commentcount integer)";
```

```
public static final String CREATE_COMMENT = "create table comment ("
 + "id integer primary key autoincrement, "
 + "content text)";

public MySQLiteHelper(Context context, String name, CursorFactory factory,
 int version) {
super(context, name, factory, version);
}

@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(CREATE_NEWS);
db.execSQL(CREATE_COMMENT);
}

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

}
```

这对于第一次安装我们软件的用户来说是完全可以正常工作的，但是如果有的用户已经安装过上一版的软件，那么很遗憾，comment表是创建不出来的，因为之前数据库就已经创建过了，onCreate()方法是不会重新执行的。

对于这种情况我们就要用升级的方式来解决了，看到MySQLiteHelper构造方法中的第四个参数了吗，这个就是数据库版本号的标识，每当版本号增加的时候就会调用onUpgrade()方法，我们只需要在这里处理升级表的操作就行了。比较简单粗暴的方式是将数据库中现有的所有表都删除掉，然后重新创建，代码如下所示：

```
public class MySQLiteHelper extends SQLiteOpenHelper {

.....

@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(CREATE_NEWS);
db.execSQL(CREATE_COMMENT);
```

```
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
db.execSQL("drop table if exists news");
onCreate(db);
}

}
```

可以看到，当数据库升级的时候，我们先把news表删除掉，然后重新执行了一次onCreate()方法，这样就保证数据库中的表都是最新的了。

但是，如果news表中本来已经有数据了，使用这种方式升级的话，就会导致表中的数据全部丢失，所以这并不是一种值得推荐的升级方法。那么更好的升级方法是什么样的呢？这就稍微有些复杂了，需要在onUpgrade()方法中根据版本号加入具体的升级逻辑，我们来试试来吧。比如之前的数据库版本号是1，那么在onUpgrade()方法中就可以这样写：

```
public class MySQLiteHelper extends SQLiteOpenHelper {

.....

@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(CREATE_NEWS);
db.execSQL(CREATE_COMMENT);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
switch (oldVersion) {
case 1:
 db.execSQL(CREATE_COMMENT);
default:
}
}

}
```

可以看到，这里在onUpgrade()方法中加入了一个switch判断，如果oldVersion等于1，就再创建一个comment表。现在只需要调用如下代码，表就可以得到创建或升级了：

```
SQLiteOpenHelper dbHelper = new MySQLiteHelper(this, "demo.db", null, 2);
SQLiteDatabase db = dbHelper.getWritableDatabase();
```

这里我们将版本号加1，如果用户是从旧版本升级过来的，就会新增一个comment表，而如果用户是直接安装的新版本，就会在onCreate()方法中把两个表一起创建了。

OK，现在软件的第二版本也发布出去了，可是就在发布不久之后，突然发现comment表中少了一个字段，我们并没有记录评论发布的时间。没办法，只好在第三版中修复这个问题了，那我们该怎么样去添加这个字段呢？主要需要修改comment表的建表语句，以及onUpgrade()方法中的逻辑，代码如下所示：

```
public class MySQLiteHelper extends SQLiteOpenHelper {

.....

public static final String CREATE_COMMENT = "create table comment ("
 + "id integer primary key autoincrement, "
 + "content text, "
 + "publishdate integer)";

.....

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
switch (oldVersion) {
case 1:
 db.execSQL(CREATE_COMMENT);
 break;
case 2:
 db.execSQL("alter table comment add column publishdate integer");
 break;
default:
}
}

}
```

可以看到，在建表语句当中我们新增了publishdate这一列，这样当执行onCreate()方法去创建表的时候，comment表中就会有这一列了。那么如果是从旧版本升级过来的呢？也没有问题，我们在onUpgrade()方法中已经把升级逻辑都处理好了，当oldVersion等于2的时候，会执行alter语句来添加publishdate这一列。现在调用以下代码来创建或升级数据库：

```
SQLiteOpenHelper dbHelper = new MySQLiteHelper(this, "demo.db", null, 3);
SQLiteDatabase db = dbHelper.getWritableDatabase();
```

将数据库版本号设置成3，这样就可以保证数据库中的表又是最新的了。

现在我们已经学习了新增表和新增列这两种升级方式，那么如果是某张表中的某一列已经没有用了，我想把这一列删除掉该怎么写呢？很遗憾，SQLite并不支持删除列的功能，对于这情况，多数软件采取的作法是无视它，反正以后也用不到它了，留着也占不了什么空间，所以针对于这种需求，确实没什么简单的解决办法。

这大概就是传统开发当中升级数据库表的方式了，虽说能写出这样的代码表示你已经对数据库的升级操作理解的比较清楚了，但随着版本越来越多，onUpgrade()方法中的逻辑也会变得愈发复杂，稍微一不留神，也许就会产生错误。因此，如果能让代码自动控制升级逻辑，而不是由人工来管理，那就是再好不过了，那么下面我们就来学习一下怎样使用LitePal来进行升级表的操作。

使用LitePal升级表

通过上一篇文章的学习，我们已经知道LitePal是一款ORM模式的框架了，已经熟悉创建表流程的你，相信对于升级表也一定会轻车熟路的。那么为了模仿传统升级表方式中的需求，现在我们也需要创建一张comment表。第一步该怎么办呢？相信你也许早就已经猜到了，那当然是先创建一个Comment类了，如下所示：

```
package com.example.databasetest.model;

public class Comment {

private int id;

private String content;

// 自动生成get、set方法
...
```

```
}
```

OK，Comment类中有id和content这两个字段，也就意味着comment表中会有id和content这两列。

接着修改litepal.xml中的配置，在映射列表中新增Coomment类，并将版本号加1，如下所示：

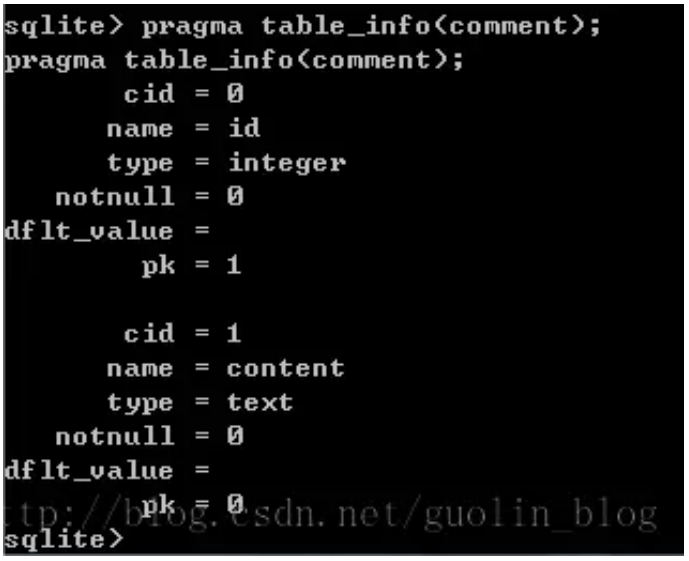
```
<?xml version="1.0" encoding="utf-8"?>
<litepal>
<dbname value="demo" ></dbname>

<version value="2" ></version>

<list>
<mapping class="com.example.databasetest.model.News"></mapping>
<mapping class="com.example.databasetest.model.Comment"></mapping>
</list>
</litepal>
```

没错，就是这么简单，仅仅两步，升级的操作就已经完成了，现在我们只需要操作一下数据库，comment表就会自动生成了，如下所示：

```
SQLiteDatabase db = Connector.getDatabase();
```

那么我们还是通过.table命令来查看一下结果，如下图所示：

OK，comment表已经出来了，那么再通过pragma命令来查看一下它的表结构吧：



没有问题，comment表中目前有id和content这两列，和Comment模型类中的字段是保持一致的。

那么现在又来了新的需求，需要在comment表中添加一个publishdate列，该怎么办呢？不用怀疑，跟着你的直觉走，相信你已经猜到应该在Comment类中添加这样一个字段了吧，如下所示：

```
public class Comment {

private int id;

private String content;

private Date publishDate;

// 自动生成get、set方法
...
}
```

然后呢？剩下的操作已经非常简单了，只需要在litepal.xml中对版本号加1就行了，如下所示：

```
<litepal>
<dbname value="demo" ></dbname>

<version value="3" ></version>
...
```

```
</litepal>
```

这样当我们下一次操作数据库的时候，publishdate列就应该会自动添加到comment表中。调用Connector.getDatabase()方法，然后重新查询comment表结构，如下所示：

可以看到，publishdate这一列确实已经成功添加到comment表中了。

通过这两种升级方式的对比，相信你已经充分体会到了使用LitePal进行升级表操作所带来的便利了吧。我们不需要去编写任何与升级相关的逻辑，也不需要关心程序是从哪个版本升级过来的，唯一要做的就是确定好最新的Model结构是什么样的，然后将litepal.xml中的版本号加1，所有的升级逻辑就都会自动完成了。LitePal确实将数据库表的升级操作变得极度简单，使很多程序员可以从维护数据库表升级的困扰中解脱出来。

然而，LitePal却明显做到了更好。前面我们提到过关于删除列的问题，最终的结论是无法解决，因为SQLite是不支持删除列的命令的。但是如果使用LitePal，这一问题就可以简单地解决掉，比如说publishdate这一列我们又不想要了，那么只需要在Comment类中把它删除掉，然后将版本号加1，下次操作数据库的时候这个列就会不见了。

那么有的朋友可能会问了，不是说SQLite不支持删除列的命令吗？那LitePal又是怎样做到的呢？其实LitePal并没有删除任何一列，它只是先将comment表重命名成一个临时表，然后根据最新的Comment类的结构生成一个新的comment表，再把临时表中除了publishdate之外的数据复制到新的表中，最后把临时表删掉。因此，看上去的效果好像是做到了删除列的功能。

这也是使用框架的好处，如果没有框架的帮助，我们显然不会为了删除一个列而大废周章地去写这么多的代码，而使用框架的话，具体的实现逻辑我们已经不用再关心，只需要控制好模型类的数据结构就可以了。

另外，如果你想删除某一张表的话，操作也很简单，在litepal.xml中的映射列表中将相应的类删除，表自然也就不存在了。其它的一些升级操作也都是类似的，相信你已经能举一反三，这里就不再赘述了。

好了，今天对LitePal的介绍就到这里吧，下篇文章当中我们会学习使用LitePal来进行表关联的操作，感兴趣的朋友请继续阅读 Android数据库高手秘籍(四)——使用LitePal建立表关联。

阅读原文

微信扫一扫
关注该公众号