#### 腾讯-数据库版本如何单独升级，并且将原有数据迁移过去

在我们开发的应用中，一般都会涉及到数据库，使用数据的时候会涉及到数据库的升级、数据的迁移、增加行的字段等。比如，用户定制数据的保存，文件的端点续传信息的保存等都会涉及到数据库。

​ 我们应用第一个版本是V1.0，在迭代版本V1.1 时，我们在数据库中增加了一个字段。因此V1.0的数据库在V1.1版本需要升级，V1.0版本升级到V1.1时原来数据库中的数据不能丢失，

​ 那么在V1.1中就要有地方能够检测出来版本的差异，并且把V1.0软件的数据库升级到V1.1软件能够使用的数据库。也就是说，要在V1.0软件的数据库的那个表中增加那个字段，并赋予这个字段默认值。 应用中怎么检测数据库需要升级呢? SQLiteOpenHelper 类中有一个方法

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

​ 当我们创建对象的时候如果传入的版本号大于之前的版本号，该方法就会被调用，通过判断oldVersion 和 newVersion 就可以决定如何升级数据库。在这个函数中把老版本数据库的相应表中增加字段，并给每条记录增加默认值即可。新版本号和老版本号都会作为onUpgrade函数的参数传进来，便于开发者知道数据库应该从哪个版本升级到哪个版本。升级完成后，数据库会自动存储最新的版本号为当前数据库版本号。

##### 数据库升级

SQLite提供了ALTER TABLE命令，允许用户重命名或添加新的字段到已有表中，但是不能从表中删除字段。并且只能在表的末尾添加字段，比如,为Orders 表中添加一个字段：”ALTER TABLE Order ADDCOLUMN Country”

代码如下：

public class OrderDBHelper extends SQLiteOpenHelper {
private static final int DB\_VERSION = 1;
private static final String DB\_NAME = "Test.db";
public static final String TABLE\_NAME = "Orders";

public OrderDBHelper(Context context, int version) {
 super(context, DB\_NAME, null, version);
}

@Override
public void onCreate(SQLiteDatabase db) {
 String sql = "create table if not exists " + TABLE\_NAME + " (Id integer primary key, " +
 "CustomName text, OrderPrice integer)";
 db.execSQL(sql);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
 Log.e("owen", "DB onUpgrade");
 if (newVersion == 2) {
 db.execSQL("ALTER TABLE " + TABLE\_NAME + " ADD COLUMN Country");
 Cursor cr = db.rawQuery("select \* from " + TABLE\_NAME, null);
 while (cr.moveToNext()) {
 String name = cr.getString(cr.getColumnIndex("CustomName"));
 ContentValues values = new ContentValues();
 values.put("CustomName", name);
 values.put("Country", "China");
 db.update(TABLE\_NAME, values, "CustomName=?", new String[] {name});
 }
 cr.close();
 }
}

OrderDBHelper orderDBHelper = new OrderDBHelper(this, 2);
SQLiteDatabase db = orderDBHelper.getWritableDatabase();

ContentValues contentValues = new ContentValues();
contentValues.put("OrderPrice", 100);
contentValues.put("CustomName", "OwenChan");
db.insert(OrderDBHelper.TABLE\_NAME, null, contentValues);

Log.e("owen", "create finish");

Cursor cr = db.rawQuery("select \* from " + OrderDBHelper.TABLE\_NAME , null);
while (cr.moveToNext()) {
 String name = cr.getString(cr.getColumnIndex("CustomName"));
 Log.e("owen", "name:" + name);
 String country = cr.getString(cr.getColumnIndex("Country"));
 Log.e("owen", "country:" + country);
}
cr.close();
db.close();

##### 数据库的迁移

可以分一下几个步骤迁移数据库

1、 将表名改成临时表

ALTER TABLE Order RENAME TO \_Order;

2、创建新表

CREATETABLE Test(Id VARCHAR(32) PRIMARY KEY ,CustomName VARCHAR(32) NOTNULL , Country VARCHAR(16) NOTNULL);

3、导入数据

INSERTINTO Order SELECT id, “”, Age FROM \_Order;

4、删除临时表

DROPTABLE \_Order;

通过以上四个步骤，就可以完成旧数据库结构向新数据库结构的迁移，并且其中还可以保证数据不会因为升级而流失。 当然，如果遇到减少字段的情况，也可以通过创建临时表的方式来实现。

实现代码如下

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
 if (newVersion == 2) {
 char str = '"';
 db.beginTransaction();
 db.execSQL("ALTER TABLE Order RENAME TO \_Order");
 db.execSQL("CREATE TABLE Order(Id integer primary key autoincrement , CustomName VARCHAR(20) NOT NULL,"
 + " Country VARCHAR(32) NOT NULL , OrderPrice VARCHAR(16) NOT NULL)");
 db.execSQL("INSERT INTO Order SELECT Id, " + str + str
 + ", CustomName, OrderPrice FROM \_Order");
 db.setTransactionSuccessful();
 db.endTransaction();
 }
}

##### 多个数据库版本的升级

假如我们开发的程序已经发布了两个版本：V1.0，V2.0，我们正在开发V3.0。版本号分别是1，2，3。对于这种情况，我们应该如何实现升级？ 用户的选择有：

1. V1.0 -> V3.0 DB 1 -> 2
2. V2.0 -> V3.0 DB 2 -> 3

数据库的每一个版本所代表的数据库必须是定义好的，比如说V1.0的数据库，它可能只有两张表TableA和TableB，如果V2.0要添加一张表TableC，如果V3.0要修改TableC，数据库结构如下：

V1.0 —> TableA, TableB V1.2 —> TableA, TableB, TableC V1.3 —> TableA, TableB, TableC (Modify)

代码如下：

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
 if (1 == oldVersion) {
 String sql = "Create table C....";
 db.execSQL(sql);
 oldVersion = 2;
 }

 if (2 == oldVersion) {
 //modify C
 oldVersion = 3;
 }
}

##### 导入已有数据库

/\*\*
 \* Created by Owen Chan
 \* On 2017-09-26.
 \*/

public class DbManager {

 public static final String PACKAGE\_NAME = "com.example.sql";
 public static final String DB\_NAME = "table.db";
 public static final String DB\_PATH = "/data/data/" + PACKAGE\_NAME;
 private Context mContext;

 public DbManager(Context mContext) {
 this.mContext = mContext;
 }

 public SQLiteDatabase openDataBase() {
 return SQLiteDatabase.openOrCreateDatabase(DB\_PATH + "/" + DB\_NAME, null);
 }

 public void importDB() {
 File file = new File(DB\_PATH + "/" + DB\_NAME);
 if (!file.exists()) {
 try {
 FileOutputStream out = new FileOutputStream(file);
 int buffer = 1024;

 InputStream in = mContext.getResources().openRawResource(R.raw.xxxx);
 byte[] bts = new byte[buffer];
 int lenght;
 while ((lenght = in.read(bts)) > 0) {
 out.write(bts, 0, bts.length);
 }
 out.close();
 in.close();
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 }
}

##### 调用方式

@Override
protected void onResume() {
 super.onResume();
 DbManager dbManager = new DbManager(this);
 dbManager.importDB();
 SQLiteDatabase db = dbManager.openDataBase();
 db.execSQL("do what you want");
}