1、懂你 个推推使用的时候，注意每次开启都要去注册

2、CusorAdapter如何独立于数据，Cusor只是一种数据的访问方式，不要局限访问内容

用户接口
contentProvider接口（创建openhelper）封装统一数据访问接口
SQLiteOpenHelper接口 （oncreate创建数据库）
数据库SQLiteDateBase接口
db存储（缓存、文件索引）
文件、缓存文件

SQLiteOpenHelper:

- SQLiteOpenHelper是SQLiteDatabse的一个帮助类，用来管理数据的创建和版本更新。一般的用法是定义一个类继承SQLiteOpenHelper，并实现两个回调方法，OnCreate(SQLiteDatabase db)和onUpgrade(SQLiteDatabse, int oldVersion, int newVersion)来创建和更新数据库。
-

Cursor

implements Closeable

---

android.database.Cursor

▸ Known Indirect Subclasses
AbstractCursor, AbstractWindowedCursor, CrossProcessCursor, CrossProcessCursorWrapper, CursorWrapper, MatrixCursor, MergeCursor, MockCursor,SQLiteCurso

Class Overview

This interface provides random read-write access to the result set returned by a database query.

Cursor implementations are not required to be synchronized so code using a Cursor from multiple threads should perform its own synchronization when using the Cursor.

- 可以看出cursor主要是在数据库的地方使用

emoji表情使用学习

Cursor理解:
本身是一个对数据表的访问itorator,主要针对数据库
CursorAdapter 数据源头是Cusor身后的那部分表格数据，View动态设定
SQliteHelper 主要是用来出创建跟更新数据库 在getWriteableDatabase()或getReadableDatabase()方法被调用之前，这个数据库不会实际的被创建。getWritableDatabase()方法可能需要很长时间才能返回，因此不应该在应用程序的主线程中调用它，包括ContentProvider.onCreate()方法。**publlic abstract void onCreate(SQLiteDatabase db)**数据库被首次创建时，会调用这个方法。这时创建数据库表和表初始化的地方。

覆盖安装的问题

```
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                NTLog.v(TAG, "Upgrading msg.db from version " + oldVersion + " to " +
newVersion);
                if (oldVersion == newVersion) {
                        return;
                }

                switch (oldVersion) {
                case 1:// add columns
                                // MsgTable(增加字段)
                        addTableColumn(db, MsgTable.TABLE_NAME, MsgTable.C_INTERGER1, INT);
                        addTableColumn(db, MsgTable.TABLE_NAME, MsgTable.C_LONG1, LONG);
                        // LastMsgTable(变更需求较少，暂时增加两个字段)
                        addTableColumn(db, LastQuestionMsgTable.TABLE_NAME,
```

```
LastQuestionMsgTable.C_STRING1, TEXT);
                        addTableColumn(db, LastQuestionMsgTable.TABLE_NAME,
LastQuestionMsgTable.C_INTERGER1, INT);
                default:
                        break;
                }
```

潜意识的找到需要优化的点，看看哪个地方可以存在优化 速度、性能、稳定性、功耗
分清楚provider与helper的关系 一个是提供操作接口，一个是管理创建与升级

App版本更新时对SQLite数据库升级或者降级遇到的问题

SQLite是Android内置的一个很小的关系型数据库。SQLiteOpenHelper是一个用来辅助管理数据库创建和版本升级问题的抽象类。我们可以继承这个抽象类，实现它的一些方法来对数据库进行自定义操作。下面两个方法必须重写：

- public void onCreate(SQLiteDatabase db)
- public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)

另外SQLiteOpenHelper子类在构造实例时必须指定当前数据库的名称（name）、版本号（version）。而这里名称就决定了数据库存储时的文件名称，而这里的版本号与App在AndroidMainfest.xml定义的versionCode没有绝对关联。也就是在App更新时如果数据库的数据结构没有发生变化那么数据库的版本号则不用增加。

onCreate：调用时机是用户首次安装应用后启动，或是清除App数据库文件后启动。这时可以在这个函数中完成初始的数据表的创建。

onUpgrade：调用时机是用户在做应用更新，覆盖安装后启动，如果新版本中数据库版本号要比旧版本中的数据库版本号高则会调用。这时可以在这个函数完成数据库版本升级带来的旧版本的兼容问题，以及数据迁移问题。

还有一个一般情况下不需要重写，但在应用出现逆向降级（如应用由版本号4降级安装版本号为3的包）时必须重写的方法**onDowngrade**，如果应用降级覆盖安装时没有重写该方法则会崩溃。

在数据库版本升级时， 我们可能会遇到这样一些情况：

- 需要扩展一个表的字段
- 删除掉原来表上某个冗余的字段
- 新建一个表

而处理上面这些问题都要在不损害旧数据库历史数据的前提下完成。这里我们假设用户手机上之前安装的是数据库版本为1的包，升级安装的是数据库版本号为2的包。这时我们要在数据库版本为2的包在去处理这些升级逻辑。

首先是扩展一个表的字段在onUpgrade中的实现为：

MyDBHelper.java

```
1  @Override
2  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
3      //旧数据库版本为1，才为表pedant添加一个student_name字段
4      if(oldVersion < 2) {
5          db.execSQL("ALTER TABLE pedant ADD COLUMN student_name text");
6      }
7  }
```

SQLite对ALTER TABLE的支持是有限的，你可以在一个存在表上添加一个字段到末尾，或者是改变表的名称。但如果你想做更复杂的操作，比如删除一个表已有的字段，就要重新创建这个表并完成数据迁移，而不能使用**DROP COLUMN**这样方便的命令了。详见SQLite Frequently Questions

比如表pedant原来有三个字段a、b、c，现在想删除c字段，那么在onUpgrade中写法如下：

MyDBHelper.java

```
1  @Override
2  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
3      //旧数据库版本为1，删除表pedant的c字段
4      if(oldVersion < 2) {
5          db.beginTransaction();
6          try {
7              db.execSQL("CREATE TEMPORARY TABLE pe_backup (a, b);");
8              db.execSQL("INSERT INTO pe_backup SELECT a, b FROM pedant;");
9              db.execSQL("DROP TABLE pedant;");
10             db.execSQL("CREATE TABLE pedant(a text, b text);");
11             db.execSQL("INSERT INTO pedant SELECT a, b FROM pe_backup;");
12             db.execSQL("DROP TABLE pe_backup;");
```

```
            db.setTransactionSuccessful();
        } finally {
            db.endTransaction();
        }
    }
}
```

这样就既完成了对c字段的删除也保留了原来表上的数据。

最后一种情况最简单直接执行CREATE语句就要可以了。

MyDBHelper.java

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    //旧数据库版本为1，创建新表newtb
    if(oldVersion < 2) {
        db.execSQL("CREATE TABLE newtb(a text, b text);");
    }
}
```

数据库在做升级时我们能明确地知道当前我们要对各旧表进行什么样的操作来兼容新版本。但如果在数据库降级时，情况就不一样了，针对我们开发新版本2时， 我们不能明确地知道以后的新版本比如版本3、4的数据库结构走向是怎样的。比如以后用户从版本3回退到我们正在开发的版本2，由于我们开发当时不能预知版本3的表结构，不知版本3的数据表能否兼容到版本2（假如版本3升级时删除了一个版本2一直在用的表字段，这时回退数据结构可能就不兼容了），那么我们在开发版本2时最稳妥的做法是**重写onDowngrade时把所有当前版本将用到的表全部重建，即降级时扔掉以前全部的数据。**

MyDBHelper.java

```
// 因为我们无法预知未来版本的表结构，向下兼容时最稳妥的方法就是将该版本自己需要的表重构一次
@Override
public void onDowngrade (SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS t1;");
    db.execSQL("DROP TABLE IF EXISTS t2;");
    db.execSQL("DROP TABLE IF EXISTS t3;");
    db.execSQL("DROP TABLE IF EXISTS t4;");
    ....
    onCreate(db); // 建表
}
```


只有多个进程间需要共享数据的时候才需要ContentProvider，当然如过你想要统一接口也可以，或者自己定义接口

A content provider is only required if you need to share data between multiple applications.

Fragment采用默认构造函数 setArgurment的方式存储变量