关系型数据库：SQLite

是一种轻型数据库管理系统，他的设计目标是嵌入式的，占用资源非常低，嵌入式设备中可能

以

特征：

轻量级 只需要几百K的内存就够了，支持高达2TB大小的数据存储

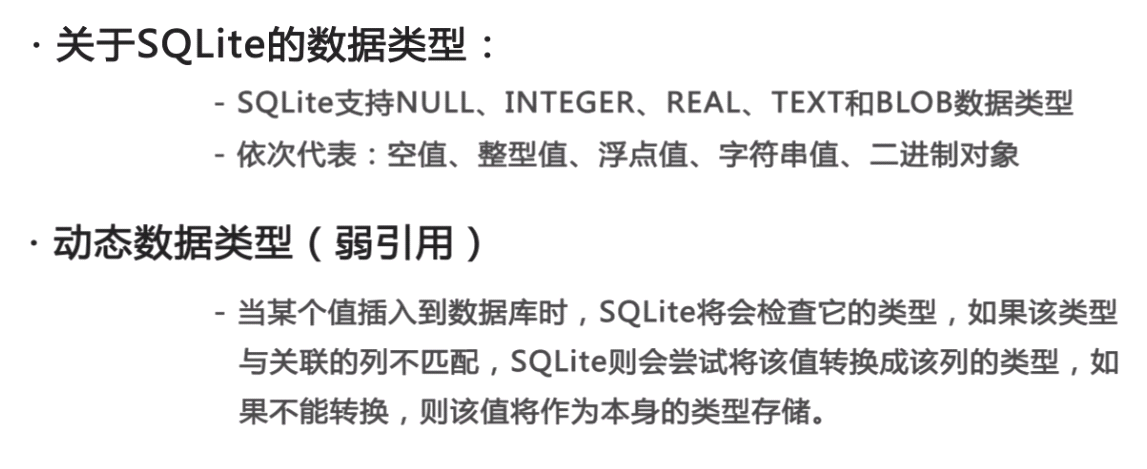
对立性 单个文件形式存在，独立 无依赖 无需安装

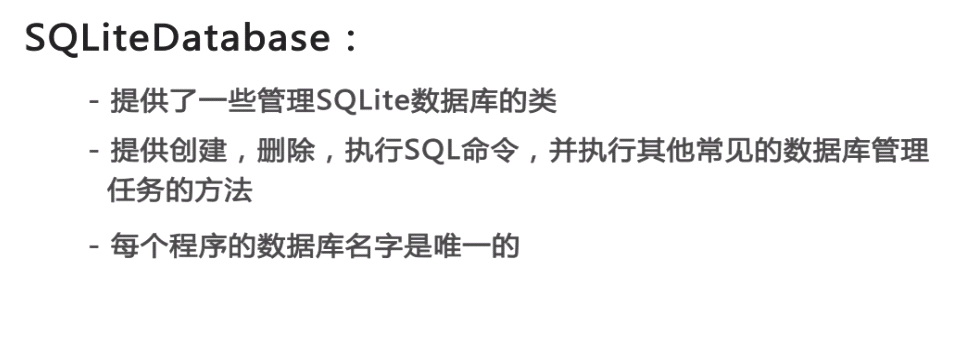
隔离性 全部在一个文件夹内

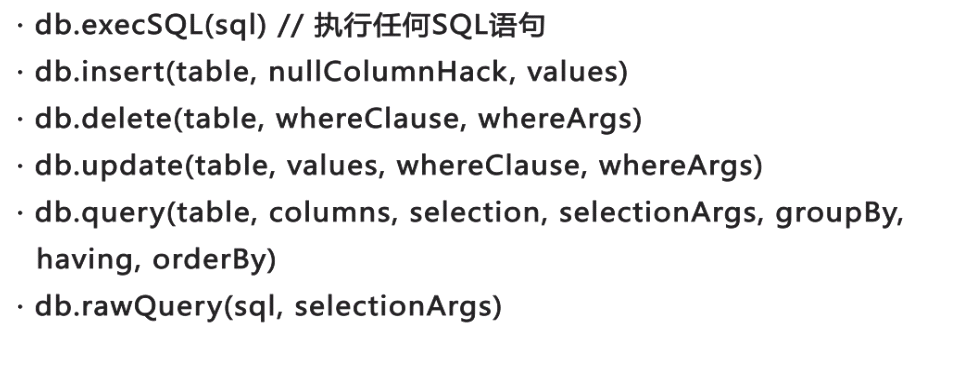
跨平台 支持众多操作系统

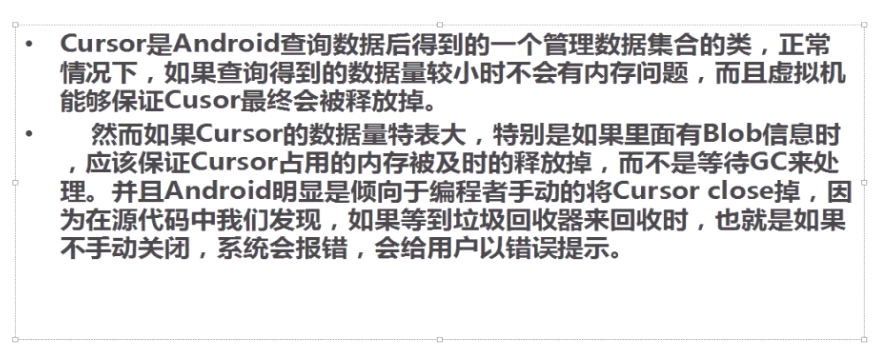
多语言接口 支持众多编程语言

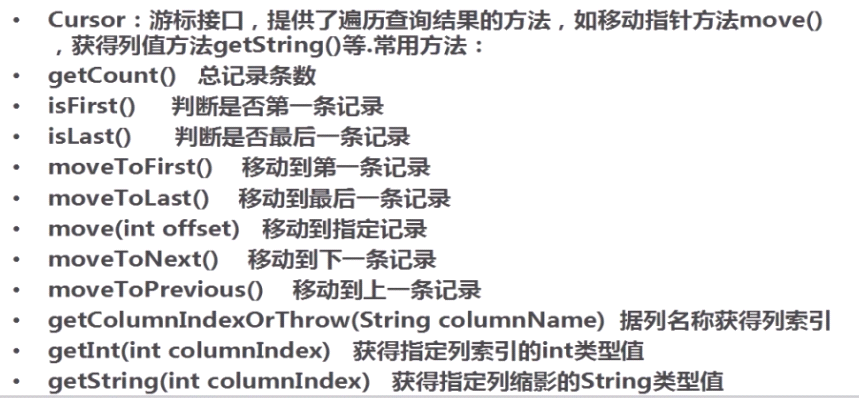
安全性 事务，通过数据库上的独占性和共享锁实现事务处理，多个进程可以在同一时间从同一数据库读取数据，但只有一个可以写入数据。











1.建立DataBase

2.建立数据连接

3.建立对象，操作数据库

4.返回值加以操作

5.关闭数据库连接

建立表

1.工具建立

2.直接代码建

public void creatDataBase()  
{  
 //建立及打开数据库，同时连接上  
 sql = context.openOrCreateDatabase(DATABASE\_NAME, Activity.*MODE\_PRIVATE*,null);  
 //建立一张表  
 //"creat TABLE users(userID INTEGER PRIMARY KEY AUTO INCREASE , userName VARCHAR(50) NOT NULL,.....)"  
 //建立表时传的String参数即SQL语句，要遵循SQL语法  
 String sqlString = "" +  
 "create TABLE "+TABLE\_NAME  
 +"("  
 +USER\_ID+" INTEGER PRIMARY KEY AUTOINCREMENT,"  
 +COLUMN\_USER\_NAME+" VARCHAR(50) NOT NULL,"  
 +COLUMN\_USER\_ADDRESS+"VARCHAR(50) NOT NULL"  
 +")";  
 sql.execSQL(TABLE\_NAME+COLUMN\_USER\_NAME+COLUMN\_USER\_ADDRESS);  
 //插入值  
 ContentValues contentValues = new ContentValues();  
 contentValues.put(COLUMN\_USER\_NAME,"Sundy");  
 contentValues.put(COLUMN\_USER\_ADDRESS,"China");  
 sql.insert(TABLE\_NAME,"",contentValues);  
 //关闭数据库连接  
 sql.close();  
 sql = null;  
}

在Activity子类中操作 默认有上下文对象 this.openOrCreateDatabase（）

*/\*\*   
 \* 原生的SQL语句操作表 不建议使用 低效率  
 \*/*public void sql01()  
{  
 SQLiteDatabase sql = openOrCreateDatabase(  
 "sql.db",//数据库名，数据文件，可以没有格式，但方便以后导出到第三方数据库工具使用  
 *MODE\_PRIVATE*, //数据库的权限  
 null //......??  
 );  
  
 //定义原生的SQL语句创建一张表  
 String sqlL =  
 "create table if not exits usertb"+  
 "(" +  
 "\_id integer primary key autoincrement,"+ //定义主键的字段名称一般以下划线开头  
 "name text not null,"+  
 "age integer not null,"+  
 "sex integer not null"+  
 ")";  
 sql.execSQL(sqlL);  
  
 //定义原生的SQL语句向表中插入一组数据  
 String inse01 ="" +  
 "insert into usertb"+  
 "(name,sex,age)"+  
 "values('张三','女',18)";  
 String inse02 ="" +  
 "insert into usertb"+  
 "(name,sex,age)"+  
 "values('李四','男',25)";  
 sql.execSQL(inse01);  
 sql.execSQL(inse02);  
  
 //定义原生的SQL语句查询  
 String quer = "select \* from usertb";  
 Cursor c = sql.rawQuery(quer,null); //游标类接收数据  
 if (c != null)  
 {  
 while(c.moveToNext())  
 {  
 //c.getInt(0);//取出当前行第一列的数组  
 int id = c.getInt(c.getColumnIndex("-id"));  
 int age = c.getInt(c.getColumnIndex("age"));  
 String name = c.getString(c.getColumnIndex("name"));  
 }  
 c.close(); //查询完将游标释放  
 }  
  
 sql.close(); //最后把数据库对象释放  
}

*/\*\*  
 \* 调用提供的方法操作数据库  
 \*/*public void sql02()  
{  
 final String TABLE\_NAME = "usetb";

//已存在则取得，未存在生成  
 SQLiteDatabase sql = openOrCreateDatabase(  
 "sql02.db",//数据库名，数据文件，可以没有格式，但方便以后导出到第三方数据库工具使用  
 *MODE\_PRIVATE*, //数据库的权限  
 null //......??  
 );  
  
  
 //定义原生的SQL语句创建一张表  
 String sqlL =  
 "create table if not exits "+TABLE\_NAME+  
 "(" +  
 "\_id integer primary key autoincrement,"+ //定义主键的字段名称一般以下划线开头(逐渐自增)  
 "name text not null,"+  
 "age integer not null,"+  
 "sex integer not null"+  
 ")";  
 sql.execSQL(sqlL);  
  
  
 //直接使用方法插入  
 ContentValues values = new ContentValues(); //类似HashMap  
 values.put("name","张三"); //key 为字段名，value 为对应类型值  
 values.put("age",25);  
 values.put("sex","男");  
 //返回值为Long类型，插入的行的id，不用就不需要接收  
 Long rowid = sql.insert(  
 TABLE\_NAME, //表名  
 null,  
 values  
 );  
 values.clear(); //清空  
 values.put("name","李四");  
 values.put("age",12);  
 values.put("sex","女");  
 sql.insert(  
 TABLE\_NAME, //表名  
 null,  
 values  
 );  
  
  
 //更新  
 values.clear();  
 values.put("sex","女");  
 sql.update(  
 TABLE\_NAME,//表名  
 values,   
 "\_id>?", //更新条件  
 new String[]{"3"} //条件  
 );  
   
   
 //删除  
 sql.delete(  
 TABLE\_NAME,  
 "name like ?",  
 new String[]{"%丰%"}  
 );  
   
}

每个程序都只有自己的数据库，默认情况下相互不干扰



@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
  
 OpenHelper oh = new OpenHelper(MainActivity.this,"daname.db");  
 SQLiteDatabase db = oh.getWritableDatabase();  
 Cursor cs = db.rawQuery("select \* from tbname",null);  
 if (cs != null)  
 {  
 String[] name = cs.getColumnNames();//得到所有字段的名字  
 while (cs.moveToNext())  
 {  
 for (String na:name)  
 {  
 //...........  
 }  
  
 }  
 }

class OpenHelper extends SQLiteOpenHelper  
{  
 */\*\*  
 \* 构造函数必须给出  
 \*/* public OpenHelper(Context context, String name)  
 {  
 super(context,name,null,1);  
 }  
  
 */\*\*  
 \* 首次创建数据库时调用，一般存放建库建表的操作  
 \** ***@param*** *db  
 \*/* @Override  
 public void onCreate(SQLiteDatabase db) {  
 //定义原生的SQL语句创建一张表  
 String sqlL =  
 "create table if not exits tbname"+  
 "(" +  
 "\_id integer primary key autoincrement,"+ //定义主键的字段名称一般以下划线开头(逐渐自增)  
 "name text not null,"+  
 "age integer not null,"+  
 "sex integer not null"+  
 ")";  
 db.execSQL(sqlL);  
 //定义原生的SQL语句向表中插入一组数据  
 String inse01 ="" +  
 "insert into usertb"+  
 "(name,sex,age)"+  
 "values('张三','女',18)";  
 db.execSQL(inse01);  
 }  
  
 */\*\*  
 \* 当版本发生改变时自动调用  
 \** ***@param*** *db  
 \** ***@param*** *oldVersion  
 \** ***@param*** *newVersion  
 \*/* @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
  
 }  
}