# **[Android数据存储之Sqlite的介绍及使用](http://www.cnblogs.com/whoislcj/p/5506294.html)**

##### **如何创建一个数据库？**

public class DBHelper extends SQLiteOpenHelper {

private static final String TAG = "DatabaseHelper";

private static final String DB\_NAME = "test\_db";//数据库名字

public static String TABLE\_NAME = "person";// 表名

public static String FIELD\_ID = "id";// 列名

public static String FIELD\_NAME= "name";// 列名

private static final int DB\_VERSION = 1; // 数据库版本

public DBHelper(Context context) {

super(context, DB\_NAME, null, DB\_VERSION);

}

/\*\*

\* 创建数据库

\* @param db

\*/

@Override

public void onCreate(SQLiteDatabase db) {

//创建表

String sql = "CREATE TABLE " + TABLE\_NAME + "(" + FIELD\_ID + " integer primary key autoincrement , " + FIELD\_NAME + " text not null);";

try {

db.execSQL(sql);

} catch (SQLException e) {

Log.e(TAG, "onCreate " + TABLE\_NAME + "Error" + e.toString());

return;

}

}

/\*\*

\* 数据库升级

\* @param db

\* @param oldVersion

\* @param newVersion

\*/

@Override

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

}

}

SQLiteOpenHelper是SQLiteDatabase的一个帮助类，主要用于操作数据库和数据库升级，一般需要自定义个DBHelper类来继承SQLiteOpenHelper，具体实现参考上面的代码。  
  
**如何创建和打开一个可以操作的数据库？**

//获取可写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

//获取可读数据库

SQLiteDatabase db = dbHelper.getReadableDatabase();

**如何添加一个数据？**

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

//生成要修改或者插入的键值

ContentValues cv = new ContentValues();

cv.put(DBHelper.FIELD\_NAME, name);

// insert 操作

db.insert(DBHelper.TABLE\_NAME, null, cv);

//关闭数据库

db.close();

#### **如何删除一条数据？**

//生成条件语句

StringBuffer whereBuffer = new StringBuffer();

whereBuffer.append(DBHelper.FIELD\_NAME).append(" = ").append("'").append(name).append("'");

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

// delete 操作

db.delete(DBHelper.TABLE\_NAME, whereBuffer.toString(), null);

//关闭数据库

db.close();

#### **如何更新一条数据？**

//生成条件语句

StringBuffer whereBuffer = new StringBuffer();

whereBuffer.append(DBHelper.FIELD\_NAME).append(" = ").append("'").append(name).append("'");

//生成要修改或者插入的键值

ContentValues cv = new ContentValues();

cv.put(DBHelper.FIELD\_NAME, name+name);

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

// update 操作

db.update(DBHelper.TABLE\_NAME, cv, whereBuffer.toString(), null);

//关闭数据库

db.close();

#### **除了以上操作SQLiteDatabase 还提供了直接执行sql语句的函数**

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

//直接执行sql语句

db.execSQL(sql);

//关闭数据库

db.close();

#### **如何查询数据呢？**

//生成条件语句

StringBuffer whereBuffer = new StringBuffer();

whereBuffer.append(DBHelper.FIELD\_NAME).append(" = ").append("'").append(name).append("'");

//指定要查询的是哪几列数据

String[] columns = {DBHelper.FIELD\_NAME};

//获取可读数据库

SQLiteDatabase db = dbHelper.getReadableDatabase();

//查询数据库

Cursor cursor = null;

try {

cursor = db.query(DBHelper.TABLE\_NAME, columns, whereBuffer.toString(), null, null, null, null);

while (cursor.moveToNext()) {

int count = cursor.getColumnCount();

String columName = cursor.getColumnName(0);

String tname = cursor.getString(0);

Log.e(TAG, "count = " + count + " columName = " + columName + " name = " +tname);

}

if (cursor != null) {

cursor.close();

}

} catch (SQLException e) {

Log.e(TAG, "queryDatas" + e.toString());

}

//关闭数据库

db.close();

 参数解说：

* table：表名称
* colums：列名称数组
* selection：条件子句，相当于where
* selectionArgs：条件语句的参数数组
* groupBy：分组
* having：分组条件
* orderBy：排序类
* limit：分页查询的限制
* Cursor：返回值，相当于结果集ResultSet

#### **如何开启事务？**

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

db.beginTransaction(); //手动设置开始事务

try{

//在此处理批量操作

for(int i =0;i<testCount;i++ ){

//生成要修改或者插入的键值

ContentValues cv = new ContentValues();

cv.put(DBHelper.FIELD\_NAME, String.valueOf(i));

// insert 操作

db.insert(DBHelper.TABLE\_NAME, null, cv);

}

db.setTransactionSuccessful(); //设置事务处理成功，不设置会自动回滚不提交

}catch(Exception e){

}finally{

db.endTransaction(); //处理完成

//关闭数据库 db.close();

}

知识扩展：

    1.)  数据库数据安全问题探讨？数据库默认存放位置data/data/pakage/database目录下，以明文存储，获取root权限之后可以任意访问。所以后续要研究一下数据库加密。

         总结的文章链接：http://www.cnblogs.com/whoislcj/p/5511522.html

    2.)数据操作之后忘记关闭数据库或者查询数据时Cursor忘记关闭引起的内存溢出。

 文章最后附上自己写的一个数据库管理类：

/\*\*

\* 数据库管理者 - 提供数据库封装

\*

\*/public class DBManager {

private static final String TAG = "DatabaseManager";

// 静态引用

private volatile static DBManager mInstance;

// DatabaseHelper

private DBHelper dbHelper;

private DBManager(Context context) {

dbHelper = new DBHelper(context.getApplicationContext());

}

/\*\*

\* 获取单例引用

\*

\* @param context

\* @return

\*/

public static DBManager getInstance(Context context) {

DBManager inst = mInstance;

if (inst == null) {

synchronized (DBManager.class) {

inst = mInstance;

if (inst == null) {

inst = new DBManager(context);

mInstance = inst;

}

}

}

return inst;

}

/\*\*

\* 插入数据

\*/

public void insertData(String name) {

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

//生成要修改或者插入的键值

ContentValues cv = new ContentValues();

cv.put(DBHelper.FIELD\_NAME, name);

// insert 操作

db.insert(DBHelper.TABLE\_NAME, null, cv);

//关闭数据库 db.close();

}

/\*\*

\* 未开启事务批量插入

\* @param testCount

\*/

public void insertDatasByNomarl(int testCount){

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

for(int i =0;i<testCount;i++ ){

//生成要修改或者插入的键值

ContentValues cv = new ContentValues();

cv.put(DBHelper.FIELD\_NAME, String.valueOf(i));

// insert 操作

db.insert(DBHelper.TABLE\_NAME, null, cv);

Log.e(TAG, "insertDatasByNomarl");

}

//关闭数据库 db.close();

}

/\*\*

\* 测试开启事务批量插入

\* @param testCount

\*/

public void insertDatasByTransaction(int testCount){

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

db.beginTransaction(); //手动设置开始事务

try{

//批量处理操作

for(int i =0;i<testCount;i++ ){

//生成要修改或者插入的键值

ContentValues cv = new ContentValues();

cv.put(DBHelper.FIELD\_NAME, String.valueOf(i));

// insert 操作

db.insert(DBHelper.TABLE\_NAME, null, cv);

Log.e(TAG, "insertDatasByTransaction");

}

db.setTransactionSuccessful(); //设置事务处理成功，不设置会自动回滚不提交

}catch(Exception e){

}finally{

db.endTransaction(); //处理完成

//关闭数据库 db.close();

}

}

/\*\*

\* 删除数据

\*/

public void deleteData(String name) {

//生成条件语句

StringBuffer whereBuffer = new StringBuffer();

whereBuffer.append(DBHelper.FIELD\_NAME).append(" = ").append("'").append(name).append("'");

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

// delete 操作

db.delete(DBHelper.TABLE\_NAME, whereBuffer.toString(), null);

//关闭数据库

db.close();

}

/\*\*

\* 删除所有数据

\*/

public void deleteDatas()

{

String sql="delete from "+ DBHelper.TABLE\_NAME;

execSQL(sql);

}

/\*\*

\* 更新数据

\*/

public void updateData(String name) {

//生成条件语句

StringBuffer whereBuffer = new StringBuffer();

whereBuffer.append(DBHelper.FIELD\_NAME).append(" = ").append("'").append(name).append("'");

//生成要修改或者插入的键值

ContentValues cv = new ContentValues();

cv.put(DBHelper.FIELD\_NAME, name+name);

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

// update 操作

db.update(DBHelper.TABLE\_NAME, cv, whereBuffer.toString(), null);

//关闭数据库 db.close();

}

/\*\*

\* 指定条件查询数据

\*/

public void queryDatas(String name){

//生成条件语句

StringBuffer whereBuffer = new StringBuffer();

whereBuffer.append(DBHelper.FIELD\_NAME).append(" = ").append("'").append(name).append("'");

//指定要查询的是哪几列数据

String[] columns = {DBHelper.FIELD\_NAME};

//获取可读数据库

SQLiteDatabase db = dbHelper.getReadableDatabase();

//查询数据库

Cursor cursor = null;

try {

cursor = db.query(DBHelper.TABLE\_NAME, columns, whereBuffer.toString(), null, null, null, null);

while (cursor.moveToNext()) {

int count = cursor.getColumnCount();

String columName = cursor.getColumnName(0);

String tname = cursor.getString(0);

Log.e(TAG, "count = " + count + " columName = " + columName + " name = " +tname);

}

if (cursor != null) {

cursor.close();

}

} catch (SQLException e) {

Log.e(TAG, "queryDatas" + e.toString());

}

//关闭数据库 db.close();

}

/\*\*

\* 查询全部数据

\*/

public void queryDatas(){

//指定要查询的是哪几列数据

String[] columns = {DBHelper.FIELD\_NAME};

//获取可读数据库

SQLiteDatabase db = dbHelper.getReadableDatabase();

//查询数据库

Cursor cursor = null;

try {

cursor = db.query(DBHelper.TABLE\_NAME, columns, null, null, null, null, null);//获取数据游标

while (cursor.moveToNext()) {

int count = cursor.getColumnCount();

String columeName = cursor.getColumnName(0);//获取表结构列名

String name = cursor.getString(0);//获取表结构列数据

Log.e(TAG, "count = " + count + " columName = " + columeName + " name = " +name);

}

//关闭游标防止内存泄漏

if (cursor != null) {

cursor.close();

}

} catch (SQLException e) {

Log.e(TAG, "queryDatas" + e.toString());

}

//关闭数据库 db.close();

}

/\*\*

\* 执行sql语句

\*/

private void execSQL(String sql){

//获取写数据库

SQLiteDatabase db = dbHelper.getWritableDatabase();

//直接执行sql语句

db.execSQL(sql);//或者

//关闭数据库 db.close();

}

}