目录

1. 如何导入外部数据库?

把原数据库包括在项目源码的 res/raw 目录下，然后建立一个 DBManager 类,读取另存为本地数据，用到的时候再读取

2. Android 中数据存储方式

Android 提供了 5 种方式存储数据：

(1) **使用 SharedPreferences 存储数据：**存储简单的参数信息，本质上是 xml.

(2) **文件存储数据：**文件存储,常用来存储大数据量的数据，但是更新麻烦。

(3) **SQLite 数据库存储数据：**当应用程序需要处理的数据量比较大

时，为了更加合理地存储、管理、查询数据，我们往往使用关系

数据库来存储数据。

(4) **使用 ContentProvider 存储数据：**一般情况下数据在各个应用中是私密的，但是因为它也是可以用来存储分享数据。

(5) **网络存储数据：**将数据放到网络云里面，然后通过网络进行访问。

注：Android 中的数据存储都是私有的，其他应用程序都是无法访问的，除非通过 ContentResolver 获取其他程序共享的数据。

3. sqlite 升级，增加字段的语句

https://blog.csdn.net/xu_song/article/details/49658195

如果遇到复杂的修改操作，比如在修改的同时，需要进行数据的转移，那么可以采取在一个事务中执行如下语句来实现修改表的需求。

(1) 将表名改为临时表：

```
ALTER TABLE Subscription RENAME TO __temp__Subscription;
```

(2) 创建新表：

```
CREATE TABLE Subscription(OrderId VARCHAR(32) PRIMARY KEY ,UserName VARCHAR(32) NOT NULL ,ProductId VARCHAR(16) NOT NULL);
```

(3) 导入数据（从更名的数据表插入到新创的表）：

```
INSERT INTO Subscription SELECT OrderId, “”,ProductId FROM __temp__Subscription;
```

或者

INSERT INTO Subscription() SELECT OrderId, “”, ProductId FROM __temp__Subscription;

* 注意 双引号”” 是用来补充原来不存在的数据的

(4) **删除临时表：** DROP TABLE __temp__Subscription;

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应为升级而流失。

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。

具体使用：

1，将表 A 重命名，改了 A_temp。

2，创建新表 A。

3，将表 A_temp 的数据插入到表 A。

```
1.  // 1, Rename table.
2.  String tempTableName = tableName + "_temp";
3.  String sql = "ALTER TABLE " + tableName +" RENAME TO " + tempTableName;
4.  execSQL(db, sql, null);
```

```
5.
6.          // 2, Create table.
7.          onCreateTable(db);
8.
9.          // 3, Load data
10.         sql =   "INSERT INTO " + tableName +
11.                 " (" + columns + ") " +
12.                 " SELECT " + columns + " FROM " + tempTableName;
13.
14.         execSQL(db, sql, null);
15.
16.         // 4, Drop the temporary table.
17.         execSQL(db, "DROP TABLE IF EXISTS " + tempTableName, null);
```

4. 数据库框架对比和源码分析

greenDAO 简介

greenDAO 是一种 Android 数据库 ORM（object/relational mapping）框架，与 OrmLite、ActiveOrm、LitePal 等数据库相比，

单位时间内可以插入、更新和查询更多的数据，而且提供了大量的灵活通用接口。

A. greenDAO 有别于其他通过反射机制实现的 ORM 框架，greenDAO 需要一个 Java 工程事先生成需要的文件，而在每一个 DAO 文件中都已经自动组装好创建和删除数据表的 sql 语句。

B. greenDAO 的增删改查方法有一些是在 Android 原生的操作方法上进行了封装，比如说上面的查询方法 queryRaw 就是对原生的 queryRaw 进行简单的封装，对于上面链式查询的最终执行也是调用了 Android 原生的查询操作。

C. 同时还有一些方法是基于 SQLiteStatement 实现的，SQLiteStatement 相比原生的 execSQL 方法还要快一些，并且最终执行时也开启了事务，性能又提升了很多。纵观整个数据插入流程，greenDAO 借助 SQLiteStatement 完成了数据的插入，避免了其他框架利用反射拼装 sql 语句而造成的执行效率低下的问题。

到时候单独列出一个 greenDao 框架图（没空列）

5. 数据库的优化

https://www.jianshu.com/p/3b4452fc1bbd

关于客户端的特殊性

1）数据非持久化

对于客户端来说数据存在不一定是为了持久化，可能是为了更好的结构合适用，甚至可以在下拉操作时将数据清除等操作。

2）更能接受冗余

客户端的开发上我们依然可以接受冗余来换取更快的查询速度。

sqlite 简介

sqlite 每个数据库都是以单个文件的形式存在，这些数据都是以 B-Tree 的数据结构形式存储在磁盘上。同时 sqlite 更改数据的时候默认一条语句就是一个事务，有多少条数据就有多少次磁盘操作。

另外还有一个很重要的就是 sqlite 的共享锁和独享锁机制，sqlite 在可以写数据库之前，它必须先读这个数据库，看它是否已经存在了.为了从数据库文件读取，第一步是获得一个数据库文件的共享锁。一个“共享”锁允许多个数据库联接在同一时刻从这个数据库文件中读取信息。“共享”锁将不允许其他联接针对此数据库进行写操作。

在修改一个数据库之前，SQLite 首先得拥有一个针对数据库文件的“Reserved”锁。Reserved 锁类似于共享锁，它们都允许其他数据库连接读取信息。单个 Reserved 锁能够与其他进程的多个共享锁一起协作。然后一个数据库文件同时只能存在一个 Reserved 。因此只能有一个进程在某一时刻尝试去写一个数据库文件。

然后将此锁提升成一个独享锁，一个临界锁允许其他所有已经取得共享锁的进程从数据库文件中继续读取数据。但是它会阻止新的共

享锁的生成。也就说，临界锁将会防止因大量连续的读操作而无法获得写入的机会。

所以从 sqlite 本身的机制看来事务的方式去提交数据，本身是多线程乃至多进程数据安全的，但是 android 在并发写的时候还是会爆出数据库锁住的异常，我们在开发过程中需要尽量避免。

基于以上的情况来看，我个人在开发中会分为以下两种情况注意性能的优化。

1、数据库性能上

1) **批量事务插入，提升数据插入的性能：**由于 sqlite 默认每次插入都是事务，需要对文件进行读写，那么减少事务次数就能简书磁盘读写次数从而获得性能提升。
2) **单条 sql 优于多条 sql：**对于几十条 sql 插入当你替换成单条 sql 时性能有所提升，但是注意的是，换成单条可读性较差，同时会出现 sql 超长的错误。
3) **读和写操作是互斥的，写操作过程中可以休眠让读操作进行：**由于第一步所说的多数据事务插入，从而会导致插入时间增长那么也会影响数据展示的速度，所以可以在插入过程中休眠操作，以便给读操作流出时间展示数据。
4) **使用索引：**适当的索引的好处是让读取变快，当然带来的影响就是数据插入修改的时间增加，因为还得维护索引的变化。不过对于大部分的读操作多于写操作的数据库来说索引还是十分有必要

的。

5) **使用联合索引：**过多的索引同时也会减慢读取的速度。对于有层级关系的关键字，就可以考虑联合索引了，比如首先根据省查询，然后根据省市查询，层层递进到省市区县的查询方式，就可以使用联合索引，效果非常好。

6) **勿使用过多索引**

7) **增加查询条件：**当你只要一条数据时增加 limit 1,这样搜索到了后面的就不会再查询了，大大的加快了速度

8) **提前将字段的 index 映射好：**减少 getColumnIndex 的时间，可以缩短一半的时间

2、数据库设计上

1) **通过冗余换取查询速度**

2) **减少数据来提升查询速度：**比如下拉操作时，先清除旧数据，再插入新数据保证数据库中的数据总量小，提升查询速度。

3) **避免大数据多表的联合查询：**一张表关联多张表，通过外键关联。用冗余来唤起查询速度了。

6. 数据库数据迁移问题

https://www.cnblogs.com/awkflf11/p/6033074.html

数据库升级，主要有以下这几种情况：

(1)增加表

(2)删除表

(3)**修改表 ：分为“增加表字段”以及“删除表字段”**

先将旧的表删除再创建新的表，这是最简单暴力的，但前面提过这不是我们想要的结果。

主要思路是：首先将原来的表进行改名称 rename table（临时表），接着创建新的表 create table，再者将旧表内的数据迁移到新表内，最后 drop table 删除临时表。

7. 谈谈对接口与回调的理解

接口回调就是指：**可以把使用某一接口的类创建的对象的引用赋给该接口声明的接口变量，那么该接口变量就可以调用被类实现的接口的方法。**实际上，当接口变量调用被类实现的接口中的方法时，

就是通知相应的对象调用接口的方法，这一过程称为对象功能的接口回调。

就好比主线程 A 开启子线程执行任务，主线程想知道子线程 B 执行得怎么样，这时候通过接口类来与子线程 b 交流。（简单来说就是 A(主线程)很忙没工夫去做一些无聊的——>这时候来了个 B(子线程),B 说:老大你这么忙,技术活你来 CV 这种活就我来吧！这时候 A 和 B 就开始疯狂输出了.但是 A 又想知道 B 到底搞定了没有.咋办？这时候电话 C(接口类)来了,当 B 做完了体力活就拿起电话 C 开始跟 A 发了个短信告诉 A 我搞定了 .然后 A 就知道 B 搞定了 开始疯狂输出下一个技术活.好了,话不多说开始撸代码.）

8. 回调的原理

https://www.jianshu.com/p/c3da97948cd0

回调函数说白了就是定义一个函数，然后通过参数传递给另一个函数调用。回调不仅是一种技术，更是一种编程思想，上面是通过回调函数来实现的，但它不仅限于回调函数，也可以用其它的技术实现(如面向对象的实现)

9. 写一个回调 demo

https://www.jianshu.com/p/c3da97948cd0

自己应该会

10. 序列化的作用，以及 Android 两种序列化的区别

java 序列化主要有 2 个作用：

(1) 对象持久化，对象生存在内存中，想把一个对象持久化到磁盘，必须已某种方式来组织这个对象包含的信息，这种方式就是序列化；

(2) 远程网络通信，内存中的对象不能直接进行网络传输，发送端把对象序列化成网络可传输的字节流，接收端再把字节流还原成对象。

Serializable 和 Parcelable 的区别

在 Android 上应该尽量采用 Parcelable，它效率更高。Parcelabe 代码比 Serializable 多一些。但速度高十倍以上。

Serializable 只需要对某个类以及它的属性实现 Serializable 接口即可，无需实现方法。缺点是使用的反射，序列化的过程较慢，这种机制会在序列化的时候创建许多的临时对象。容易触发 GC。

Parcable 方法实现的原理是将一根完整的对象进行分解，而分解后的每一部分都是 Intent 所支持的数据类型，这样也就实现传递对象的功能。

11. 差值器（插值器）

https://www.jianshu.com/p/f18517076b40

根据事件流逝的百分比 计算 当前属性改变的百分比.举个例子,就是根据你设置的 Durantion 流逝的百分比,来改变位移(tanslate)的速度.

场景

实现非线性运动的动画效果（非线性运动：动画改变的速率不是一成不变的，如加速 & 减速运动都属于非线性运动）

目前,Android 中已经有了几种插值器,如下:

1) TimeInterpolator：一个父接口,所有 Interpolator 都应该实现它.其中方法中的 input 值 是百分比, 也就是取值在 0~1 之间.

2) AccelerateDecelerateInterpolator 在动画开始与结束的地方速率改变比较慢，在中间的时候加速

3) AccelerateInterpolator 在动画开始的地方速率改变比较慢，然后开始加速

4) AnticipateInterpolator 开始的时候向后甩一点然后向前

5) AnticipateOvershootInterpolator 开始的时候向后甩一点然后向前超过设定值一点然后返回

6) BounceInterpolator 动画结束的时候弹起，类似皮球落地

7) CycleInterpolator 动画循环播放特定的次数回到原点，速率改变沿着正弦曲线

8) DecelerateInterpolator 在动画开始的地方快然后慢

9) LinearInterpolator 以常量速率改变

10) OvershootInterpolator 向前超过设定值一点然后返回

12. 估值器

https://www.jianshu.com/p/f18517076b40

设置 属性值 从初始值过渡到结束值 的变化具体数值。插值器（Interpolator）决定 值 的变化规律（匀速、加速 blabla），即决定的是变化趋势；

而具体变化数值则交给估值器 evaluator.

fraction：动画完成度,插值器 getInterpolation（）的返回值(input 的值决定了 fraction 的值),代表时间流逝的百分比

startValue：动画的初始值

endValue：动画的结束值

常见的实现类

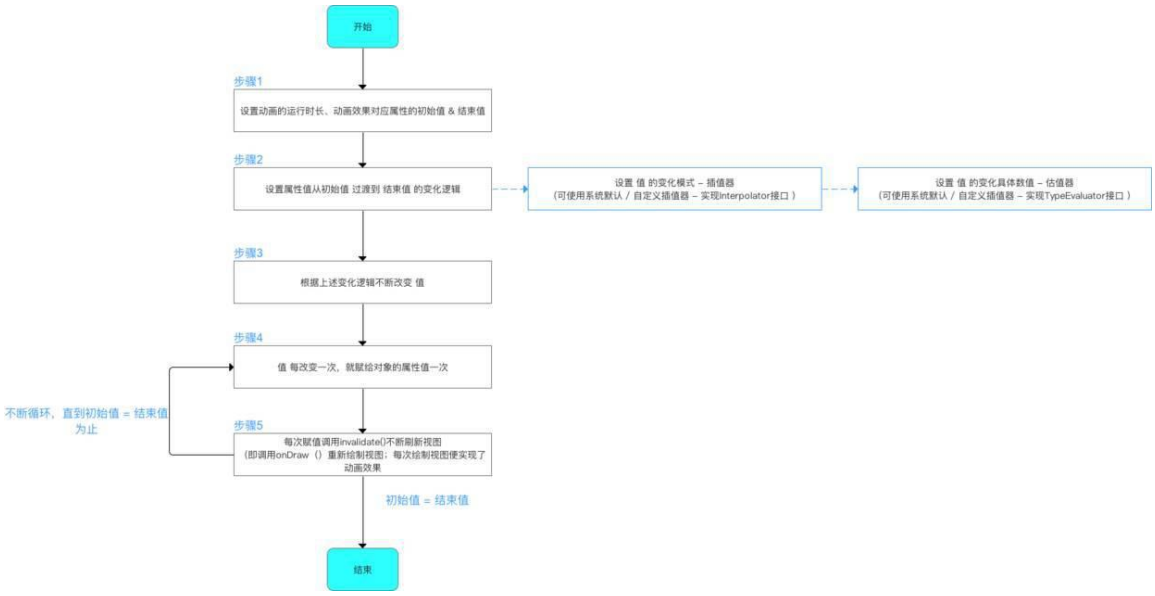
IntEvaluator Int 类型估值器，返回 int 类型的属性改变

FloatEvaluator Float 类型估值器，返回 Float 类型属性改变

ArgbEvaluator 颜色类型估值器

插值器的 input 值 和 估值器 fraction 关系

input 的值决定了 fraction 的值：input 值经过计算后传入到插值器的 getInterpolation（），然后通过实现 getInterpolation（）中的逻辑算法，根据 input 值来计算出一个返回值，而这个返回值就是 fraction 了} }



13. Android 为什么引入 Parcelable？（原理）

https://blog.csdn.net/jaycee110905/article/details/21517853

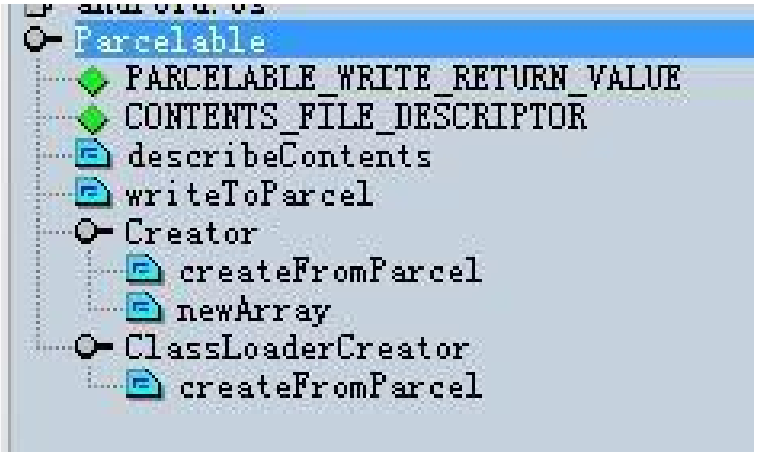
Serializable 会使用反射，序列化和反序列化过程需要大量 I/O 操作， Parcelable 自己实现封送和解封（marshalled &unmarshalled）操作不需要用反射，数据也存放在 Native 内存中，效率要快很多。

Parcelable 也是一个接口，只要实现这个接口，一个类的对象就可以实现序列化并可以通过 Intent 和 Binder 传递(除基本类外的类数据）

Parcelable 主要用在内存序列化上，通过 Parcelable 将对象序

列化到存储设备中或者将对象序列化后通过网络传输也都是可以的

Parcelable 的定义：



（1）public int describeContents()：写入接口函数，用来打包

（2）

public void writeToParcel(Parcel dest, int flags)：读取接口，目的是从 parcel 中构造一个实现了 parcelable 的类的实例对象，因为实现类这里是不可知的，所以需要用到模板的方法，继承类通过模板参数传入。

为了能够实现模板参数的传入，定义了 creator 嵌入接口，内涵两个接入函数分别是单个和多个继承类实例。

（3）public interface Creator<T> {

```
public T createFromParcel(Parcel source);
public T[] newArray(int size);
```

}

还有一个子接口继承 Creator，子接口只提供了一个函数，返

回单个继承类实例

```
public interface ClassLoaderCreator<T> extends
Creator<T>
```

Parcelable 的实现使用：

Parcelabel 的实现，需要在类中添加一个静态成员变量 CREATOR，这个变量需要继承 Parcelable.Creator 接口。

.

```
package com.zlc.provider;


import android.os.Parcel;
import android.os.Parcelable;


public class Students implements Parcelable{
    private int stu_id;
    private String stu_name;
    public Students(Parcel source){
        stu_id = source.readInt();
        stu_name = source.readString();
    }
    public int getStu_id() {
        return stu_id;
```

```
    }
    public void setStu_id(int stu_id) {
        this.stu_id = stu_id;
    }
    public String getStu_name() {
        return stu_name;
    }
    public void setStu_name(String stu_name) {
        this.stu_name = stu_name;
    }
    @Override
    public int describeContents() {
        // TODO Auto-generated method stub
        return 0;
    }
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        // TODO Auto-generated method stub
        dest.writeInt(stu_id);
        dest.writeString(stu_name);
    }
    //Interface that must be implemented and provided as a public
CREATOR field that generates instances of your Parcelable class from a
Parcel.
    public final static Parcelable.Creator<Students> CREATOR = new
Parcelable.Creator<Students>() {
```

```
        @Override
        public Students createFromParcel(Parcel source) {
            // TODO Auto-generated method stub
            return new Students(source);
        }

        @Override
        public Students[] newArray(int size) {
            // TODO Auto-generated method stub
            return new Students[size];
        }
    };
}
```

Parcelable 和 Serializable 的区别：

android 自定义对象可序列化有两个选择一个是 Serializable 和 Parcelable

一、对象为什么需要序列化

1. 永久性保存对象，保存对象的字节序列到本地文件。

2. 通过序列化对象在网络中传递对象。

3. 通过序列化对象在进程间传递对象。

二、当对象需要被序列化时如何选择所使用的接口

1.在使用内存的时候 Parcelable 比 Serializable 的性能高。

2.Serializable 在序列化的时候会产生大量的临时变量，从而引起频繁的 GC（内存回收）。

3.Parcelable 不能使用在将对象存储在磁盘上这种情况，因为在外界的变化下 Parcelable 不能很好的保证数据的持续性。

14. 有没有尝试简化 Parcelable 的使用？（原理）

as 的插件

https://blog.csdn.net/csdnzouqi/article/details/78617235

(1)在 Android studio 上下载 android parcelable code generator 插件

(2)点击右键弹出提示框，选择 Parcelable 生成即可：

(3)序列化时选择需要的属性：

(4)最后是自动生成的代码，也表示成功的实现了 Parcelable 接口：