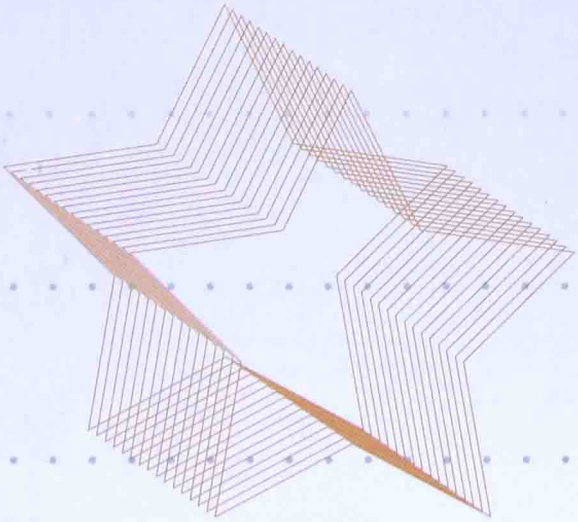
Android Database Programming

Android数据库程序设计

（美）Jason Wei ◎著
王学昌　吴骅　林展宏 ◎译



清华大学出版社

Android 数据库程序设计

（美）Jason Wei 著

王学昌 吴 骅 林展宏 译

清华大学出版社

北 京

内 容 简 介

本书详细阐述了与 Android 数据库程序设计相关的基本解决方案，主要包括在 Android 系统中存储数据、使用 SQLite 数据库、SQLite 查询、使用 Content Providers、表查询操作、UI 关联、Android 数据库应用、外部数据库、数据的收集与存储以及综合示例等内容。此外，本书还提供了相应的示例、代码，以帮助读者进一步理解相关方案的实现过程。

本书适合作为高等院校计算机及相关专业的教材和教学参考书，也可作为相关开发人员的自学教材和参考手册。

北京市版权局著作权合同登记号 图字：01-2013-6531

图书在版编目（CIP）数据

Android 数据库程序设计/（美）魏（Wei, J.）著；王学昌，吴骅，林展宏译. —北京：清华大学出版社，2014

书名原文：Android database programming

ISBN 978-7-302-37846-4

I. ①A… II. ①魏… ②王… ③吴… ④林… III. ①移动终端-应用程序-程序设计
IV. ①TN929.53

中国版本图书馆 CIP 数据核字（2014）第 199319 号

责任编辑：钟志芳
封面设计：刘　超
版式设计：文森时代
责任校对：王　云
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×230mm　　**印　　张**：11.5　　**字　　数**：238 千字
版　　次：2014 年 12 月第 1 版　　**印　　次**：2014 年 12 月第1 次印刷
印　　数：1～4000
定　　价：49.00 元

产品编号：053948-01

译 者 序

本书由浅入深地介绍了 Google 为 Android 选择的 SQLite 数据库的一些内容，其中不仅包含了最基本的 SQLite 查询，还包含了具体的操作实例，从本地存储到外部数据，数据存储和收集以及数据库应用等内容。最后还通过实例展示一些实际的应用案例。

本书的目的是使用 Google 已经内置在 Android 操作系统当中的多种方法来探索数据以及 Android 系统本身。本书不仅力求让读者了解不同的数据存储方法，还会比较每一种方法的优缺点。最后，希望通过这本书，用户能够高效创建、精心设计一个可扩展的数据中心应用程序。

本书的内容很实用，可以作为 Android 数据库程序设计的参考，不论是初学者还是已经有一定的 Android 数据库开发经验，需要进行 Android 数据库程序设计的开发人员，都能够从本书中找到所需要的知识。

本书的翻译由王学昌组织完成，参与本书翻译的还有吴骅、林展宏、周娟、刘红军、王玲、郑正正、秦双夏、莫鸿强、李远明、陶日然、黄善斌等人，感谢这些同行。由于水平有限，译文中的不当之处在所难免，恳请同行及各位读者朋友不吝赐教。

译 者

前　言

今天，我们生活在一个越来越多地依赖以数据为中心和数据驱动的世界中。如亚马逊之类的公司对用户查看和购买的商品进行信息跟踪，以便能够向用户推荐更多类似产品；如 Google 公司，存储通过它搜索的每一条查询，以便在未来能够提供更好的搜索查询建议；类似 Facebook 这样的社交媒体网站会记录用户与朋友之间的每一个事件，以便更好地了解数以百万计的用户。我们生活在以数据为中心的世界，开发以数据为中心的应用程序正是我们的当务之急。

在过去几年中，智能手机和平板电脑等移动设备的使用量一直呈爆炸式的增长。本书的目的是使用 Google 已经内置在 Android 操作系统当中的多种方法来探索数据以及 Android 系统本身。本书不仅力求让读者了解不同的数据存储方法，还会比较每一种方法的优缺点。最后，希望通过本书，用户能够高效创建、精心设计一个可扩展的数据中心应用程序。

本书的结构

第 1 章（在 Android 系统中存储数据）：侧重于 Android 系统上所有可用的各种本地数据存储方法。每个存储方法都会附带较多的代码示例，以及优缺点的比较。

第 2 章（使用 SQLite 数据库）：通过介绍自定义 SQLite 数据库的实现深入了解最复杂和最常用的本地数据存储形式——SQLite 数据库。

第 3 章（SQLite 查询）：粗略介绍 SQL 查询语言。指导读者如何构建与任意 SQLite 数据库一起使用的功能强大的数据库查询。

第 4 章（使用 Content Providers）：扩展前面几章 SQLite 数据库的内容，向读者介绍如何使用 Content Provider 将其数据库共享给 Android 操作系统。介绍一个 Content Provider 的完整实施过程，并讨论了数据公开化的好处。

第 5 章（表查询操作）：深入探讨 Android 操作系统所提供的最为广泛的 Content Provider 应用——Contacts。探讨了 Contacts 表的结构，并提供常见查询的实例。

第 6 章（UI 关联）：讨论用户将数据关联到用户界面的方法。因为数据以列表形式显示，因此本章的实例介绍了两种典型的列表适配器。

第 7 章（Android 数据库应用）：尝试避开程序设计，专注于更高的设计理念。本章

讲述到目前为止所有可以使用的本地存储方法，并且强调这些本地方法的不足之处。由此接下来的几章介绍的都是外部存储方面的内容。

第 8 章（外部数据库）：介绍使用外部数据库的概念，并向读者介绍几种常见的外部数据存储。最后介绍如何创建 Google App Engine 数据存储的实例。

第 9 章（数据的收集与存储）：通过介绍应用程序收集可插入外部数据库数据的方法，用以扩展前面章节的内容。收集数据的方法包括使用可用的 API，也包括编写自定义的 Web 抓取器模块。

第 10 章（综合示例）：本章将会完成第 8 章和第 9 章当中开始编写的应用程序。其中涉及如何首先创建 HTTP servlet，其次从移动应用程序中发起对这些 HTTP servlet 的 HTTP 请求。本章内容将作为全书的最终目标，向读者介绍如何将移动应用程序连接到他们的外部数据库，最后进行解析并以列表形式显示 HTTP 响应。

阅读本书所需基础

阅读本书需要了解 Android 操作系统的基本知识、一个可以创建 Android 和 Google App Engine 项目的程序开发集成环境（如 Eclipse 等），以及需要一个可以完成基本 Web 请求的互联网连接。

本书适合的读者

本书的读者对象是具有一定数据库经验和其他后端设计概念，并且想要了解移动应用程序中如何应用这些概念的开发人员。具有移动应用程序开发和（或）Android 平台经验，但对后端系统和设计、实施数据库架构都不太熟悉的开发人员，可能会觉得这本书很有用。

即使对于已经具有丰富的 Android 程序设计和数据库实施经验的人员，本书可能有助于他们进一步巩固概念，并了解 Android 上更多的数据存储方法。

体例

在本书中，将会发现一些用来区分不同的信息所使用不同样式的字体。以下是这些样式的例子，并附有说明。

在文本中的代码单词表示如下：“想要转换为字节形式的字符串，传递到输出流的 write()方法”。

代码块的样式设置如下：

```
Set<String> values = new HashSet<String>();
values.add("Hello");
values.add("world");
Editor e = sp.edit();
e.putStingSet("strSetKey", values);
e.commit();
Set<String> ret = sp.getSeringSet(values, new HashSet<String>());
For(String r : ret) {
    Log.i("SharedPreferencesExample", "Retrieved vals: " + r);
}
```

当需要引起对特殊代码段的注意时，我们将相关的行或条目设置为粗体显示，如下所示：

```
<uses-sdk android:minSdkVersion="5" />
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
```

命令行内的输入或者输出如下：

```
adb-s emulator-xxxx shell
```

新术语和重要的字眼以粗体显示。

读者反馈

我们一直希望得到读者的回馈，这可以让我们了解用户对于这本书的看法，如喜欢什么而不喜欢什么。读者反馈有助于我们改进本书的内容，而这将会获得更大的收获。

可以通过发送 E-mail 至 feedback@packtpub.com 提交反馈信息，请务必在邮件主题当中提及本书的书名信息。

如果你在某一领域内有专长，并且有兴趣将其写成书或参与书的编写，请查阅网站 www.packtpub.com/authors 的作者指南信息。

用户支持

现在，读者已经是 Packt 图书的用户，我们有许多方式可以满足你的需求。

本书源代码下载

可以在 http://www.packtpub.com 站点下载所购买的所有 Packt 出版书籍中的实例代码。如果在其他地方购买了本书，可以访问 http://www.packtpub.com/support 页面进行注册，并通过电子邮件完成注册。

勘误表

尽管我们非常仔细地确保内容的准确性，但错误在所难免。如果读者发现其中的错误（文本错误或者代码错误）并将错误反馈给我们，我们将非常感激。这样，可以防止其他人阅读到这个错误，并帮助我们在本书的后续版本中改进。如果你发现了任何勘误内容，请访问 http://www.packtpub.com/support，选择对应书籍，单击链接 errata submission form，输入具体的勘误内容。一旦勘误内容得到确认，读者提交的内容将会上传至我们的网页或者增加到现有的勘误表中（列在勘误表一节的末尾）。在 http://www.packtpub.com/support 中可以通过选择主题查看现有的勘误表信息。

版权声明

互联网上一直面临着版权被盗版侵权的问题。Packt 一直对版权内容予以保护。如果读者在互联网上遇见了我们作品的任何一种非法副本，请立即向我们提供地址和网站名称，以便我们做出相应处理。

请通过 copyright@packtpub.com 向我们反映涉嫌盗版材料的链接。

我们真心感谢你对保护作者所提供的帮助，我们将有能力为你提供更有价值的内容。

疑难解答

在阅读本书的过程中遇到任何问题，可以通过电子邮件 questions@packtpub.com 进行咨询，我们会尽力解决这些问题。

作者简介

Jason Wei 2011 年毕业于斯坦福大学，在校期间他获得了数学计算科学专业的理学学士，同时辅修了统计学专业，接着获得了管理科学与工程专业的理学硕士，主要研究方

向为机器学习。在大学期间的前两年里，他在硅谷进行第一次创业，而在他第二次创业（BillShrink 公司）期间开始接触到 Android 系统。

从那以后，他开发了一些应用程序，从小屏幕上的娱乐程序到金融定价与建模工具等。他喜欢使用 API 进行开发，并参加了一些应用程序开发竞赛——赢得了包括 Google、MyGengo、IndexTank 等公司的奖项。除了开发应用程序之外，Jason 还喜欢编写 Android 教程，并将自己的开发经验分享在他的博客当中（thinkandroid.wordpress.com）。他在博客当中所获得的成就，使他第一次成为了《*Learning Android Game Programming*》一书的技术审校。

目前，Jason 在纽约从事量化交易员工作。

关于技术审校

Joseph Lau 目前是斯坦福大学的在读研究生，攻读计算机科学硕士学位。暑假期间，他在 LinkedIn 和 Google 的各类技术职位做实习生。他的业余爱好是 Android 程序设计，并且编写了几个 Android 应用程序。他认为移动应用程序是 21 世纪技术创新的关键部分，他还认为如果目前还没有学习 Android 程序设计，那么现在就是去接触的最好时机。

Prashant Thakkar（Pandhi）具有 7 年以上的 IT 从业经验。擅长 Java，类 Struts、Hibernate 的 J2EE 框架，以及相关的开放源代码框架。Prashant 致力于 Android 已经两年多，并且还交付过关键业务企业移动应用程序。他还对在云端交付应用程序的 Google App Engine 感兴趣。Prashant 在他的两个博客中记录了他的技术经验，分别是 http://ppandhi.wordpress.com 和 http://androidpartaker.wordpress.com。

目　录

第 1 章　在 Android 系统中存储数据

今天，我们生活在一个越来越多地依赖以数据为中心和数据驱动的世界当中。如亚马逊之类的公司对用户查看和购买的商品进行信息跟踪，以便能够向用户推荐更多类似产品；如 Google 这样的公司，存储通过它搜索的每一条查询，以便在未来能够提供更好的搜索查询建议；类似 Facebook 这样的社交媒体网站会记录用户与朋友之间的每一个事件，以便更好地了解数以百万计的用户。我们生活在以数据为中心的世界，开发以数据为中心的应用程序正是我们的当务之急。

读者可能会问，为什么会是 Android？或者再通俗点，为什么会是移动应用程序？在过去几年中，智能手机和平板电脑等移动设备使用量一直呈爆炸式的增长。此外，移动设备也隐含地给予在桌面应用程序当中所不具备的另一层面上的数据。当随身携带智能手机或平板电脑时，会通过这些设备得到当前所处的位置，也能够知道在何处办理手续或者在做些什么事情。简而言之，它所了解的可能比人们所能够意识到的更多。

记住以上两点之后，现在使用 Google 已经内置在 Android 操作系统当中的多种方法来探索数据以及 Android 系统本身。本书假定读者都已经具有一些 Android 操作系统方面的相关经验，因此在这里将会深入到具体的代码层面上。但同样重要的是读者要知道所有可用的各种数据存储方法，还需要理解每一种方法的优劣，这样才能够创建高效的、易于设计和可伸缩的应用程序。

1.1　使用 SharedPreferences

SharedPreferences 是 Android 应用程序中存储本地数据最简单、快速和高效的方法。它的本质是一个框架，允许用户存储并关联每一个 key-value（键-值）对到用户的应用程序（可以认为是映射到应用程序，以便用户可以随时访问）。此外，由于每个应用程序都与自己的 SharedPreferences 类相关联，因此被存储以及提交的数据将会在所有用户会话当中得以体现。然而，由于其简单和高效的性质，SharedPreferences 只允许用户存储原始数据类型，即 boolean（布尔型）、float（浮点型）、long（长整型）、int（整型）和 string（字符型），因此在决定使用 SharedPreferences 存储数据时要牢记这一点。

下面是如何访问和使用应用程序的 SharedPreferences 类的一个例子。

```
public class SharedPreferencesExample extends Activity {
    private static final String MY_DB = "my_db";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //INSTANTIATE SHARED PREFERENCES CLASS
        SharedPreferences sp = getSharedPreferences(MY_DB,
        Context.MODE_PRIVATE);
        //LOAD THE EDITOR - REMEMBER TO COMMIT CHANGES!
        Editor e = sp.edit();
        e.putString("strKey", "Hello World");
        e.putBoolean("boolKey", true);
        e.commit();
        String stringValue = sp.getString("strKey", "error");
        boolean booleanValue = sp.getBoolean("boolKey", false);
        Log.i("LOG_TAG", "String value: " + stringValue);
        Log.i("LOG_TAG ", "Boolean value: " + booleanValue);
    }
}
```

首先来分析这段代码：在这里首先启动一个Activity；此外，在onCreat()方法中，请求检索Shared Preferences类。getSharedPreference()方法的语法是：

```
getSharedPreferences(String mapName, int mapMode)
```

其中第一个参数指明了要映射的 shared preference（每一个应用程序可以拥有几个单独的 shared preference，因此，就像在数据库中指定表名一样，必须指明要检索的映射）。在上面的例子当中的第二个参数有点复杂，将 MODE_PRIVATE 作为参数，该参数只需指定要检索的 shared preference 实例的可见性（在例子中可见性设置为私有，这样的话只有当前的应用程序才能访问映射的内容）。其他模式如下。

- MODE_WORLD_READABLE：映射的内容其他应用程序可见，但只读。
- MODE_WORLD_WRITEABLE：其他应用程序可对映射的内容进行读写。
- MODE_MULTI_PROCESS：该模式从 API Level 11（Android 3.0 版本）开始提供，允许用户使用多个进程修改，可能会写入相同 shared preference 实例的映射。

现在，只要有了 shared preference 对象，就可以立即使用各种 get()方法对内容进行检索。例如前面提到的 getString()和 getBoolean()。get()方法通常有两个参数：第一个为 key 关键字，第二个参数在如果没有找到给定 key 关键字的情况下，则为默认值。在之前的那个例子当中，就是用了如下的 get()方法：

```
String stringValue = sp.getString("strKey", "error");
boolean booleanValue = sp.getBoolean("boolKey", false);
```

因此，在第一种情况下所要检索的是与关键字 strKey 关联的字符，如果没有相关联的关键字，那么默认的字符串为 error。类似地，在第二种情况下要检索与 boolKey 关键字相关联的布尔值，如果没有这样的键值存在，那么布尔值默认为 false。

然而，如果要编辑（edit）内容或添加新的（add new）内容，还要检索每个 shared preference 实例包含的 Editor 对象。这个 Editor 对象包含所有允许用户传递键及其相关值（例如标准 Map 对象）的 put()方法。唯一需要注意的是，在添加或更新了 shared preference 的内容后，需要调用 Editor 对象的 commit()方法来落实更新。此外，就像一个标准的 Map 对象，Editor 类也包含 remove()和 clear()方法，使用户可以自由操作 shared preference 的内容。

在介绍使用 SharedPreferences 的典型用例之前，注意如果要将 shared preference 实例的可见性设置为 MODE_WORLD_WRITEABLE，那么就可能会将自身暴露于那些有很多安全漏洞的众多外部恶意应用程序当中。因此在实际应用中并不推荐此模式。然而众多开发人员面临在两个应用程序间共享信息的需求，其解决的方法是在所涉及的应用程序清单文件中设置 android:sharedUserID。

它的工作原理是，每个应用程序在签名和导出时，会自动生成应用程序 ID。如果在应用程序清单文件中明确设置了这个 ID，假设两个应用程序使用相同的密钥进行签名，那么它们就不需要将其数据暴露给用户手机上其他应用程序能够自由地访问彼此的数据。换句话说，当两个应用程序设置了相同的 ID，这两个（也只有这两个）应用程序可以访问彼此的数据。

1.2　SharedPreferences 的常见使用案例

现在已经知道如何实例化以及编辑一个 shared preference 对象，对这种类型的数据存储需要考虑它的典型用例。

接下来的几个例子将会展示应用程序所保存的那些小型的、原始键-值数据对的类型。

1.2.1　检查用户是不是第一次访问应用程序

大多数应用程序在用户第一次访问它时，会显示一些说明/向导或者启动画面的 activity，如下所示：

```
public class SharedPreferencesExample2 extends Activity {
    private static final String MY_DB = "my_db";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        SharedPreferences sp = getSharedPreferences(MY_DB,
        Context.MODE_PRIVATE);
        /**
        * CHECK IF THIS IS USER'S FIRST VISIT
        */
        boolean hasVisited = sp.getBoolean("hasVisited", false);
        if (!hasVisited) {
            //...
            //SHOW SPLASH ACTIVITY, LOGIN ACTIVITY, ETC
            //...
            //DON'T FORGET TO COMMIT THE CHANGE!
            Editor e = sp.edit();
            e.putBoolean("hasVisited", true);
            e.commit();
        }
    }
}
```

1.2.2　应用程序最后一次更新时进行检查

许多应用程序会有一些缓存或者同步、内建的功能等都可能会要求进行常规更新。要节省更新时间，可以快速检测距离上次更新已过去的时间，然后决定是否需要进行更新或同步，代码如下：

```
/**
* CHECK LAST UPDATE TIME
*/
long lastUpdateTime = sp.getLong("lastUpdateKey", 0L);
long timeElapsed = System.currentTimeMillis() -
lastUpdateTime;
//YOUR UPDATE FREQUENCY HERE
final long UPDATE_FREQ = 1000 * 60 * 60 * 24;
if (timeElapsed > UPDATE_FREQ) {
//...
//PERFORM NECESSARY UPDATES
//...
```

```
}
//STORE LATEST UPDATE TIME
Editor e = sp.edit();
e.putLong("lastUpdateKey", System.currentTimeMillis());
e.commit();
```

下载本书例子的代码

用户可以在 Packt 网站 http://www.PacktPub.com 下载以用户本人账号所购买的书籍中的实例代码。如果是在别处购买了本书，可以访问 http://www.PacktPub.com/support 并填写相关信息，将有电子邮件指引用户进行注册。

1.2.3　保存用户登录用户名

一些应用程序允许记住用户自己的用户名（例如 PIN、电话号码等其他面向登录的字段），使用 shared preference 是存储简单原始字符 ID 的好方法，代码如下：

```
/**
* CACHE USER NAME AS STRING
*/
//TYPICALLY YOU WILL HAVE AN EDIT TEXT VIEW
//WHERE THE USER ENTERS THEIR USERNAME
EditText userNameLoginText = (EditText)
findViewById(R.id.login_editText);
String userName =
  userNameLoginText.getText().toString();
Editor e = sp.edit();
e.putString("userNameCache", userName);
e.commit();
```

1.2.4　保存应用程序的状态

有些应用程序的功能可能会根据应用程序的状态发生变化。例如设置电话铃声的应用程序，当用户指明在静音时不启动任何程序，那么这个重要的状态需要被保存下来，如下所示：

```
/**
* REMEBERING A CERTAIN STATE
*/
```