[SQLite的使用场景](http://blog.csdn.net/program_think/article/details/4107369)

分类： [开源项目](http://blog.csdn.net/program_think/article/category/513187)2009-04-24 15:47 4113人阅读 [评论](http://blog.csdn.net/program_think/article/details/4107369#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/program_think/article/details/4107369#report)

　　上次针对SQLite进行了扫盲，之后有同学在[评论里](http://program-think.blogspot.com/2009/03/opensource-review-sqlite-database.html" \l "comments) 问俺：如何在项目中使用它？今天咱来聊一下这个话题。  
  
　　★**如何权衡？**   
　　当你在权衡某个场合是否应该使用SQLite时，（在技术层面）至少要考虑如下几点：  
　　◇能否发挥SQLite的某些特长？  
　　◇是否还有其它的替代方案？  
　　◇是否有啥潜在的技术风险？  
　　想清楚上述问题之后，再做出决策。  
  
　　★**SQLite的特点**   
　　关于SQLite的特长，在[上次的帖子](http://program-think.blogspot.com/2009/03/opensource-review-sqlite-database.html" \t "_blank) 中已经介绍过了。考虑到某些同学比较健忘，咱再回顾一下：  
　　◇文件型数据库，且只有单一数据文件  
　　◇轻量级  
　　◇绿色（不依赖其它软件库）  
　　◇跨平台（包括引擎和数据文件）  
　　◇支持内存数据库  
　　◇支持较大的数据文件（TB级别）  
  
　　★**可能的替代方案**   
　　刚才说了，权衡SQLite的使用需要考虑其它的替代方案，所以俺简单介绍一下和SQLite用途相近的其它几种技术手段。后面讲应用场景的时候，会结合这几个替代方案来作对比。  
  
　　◇[Access数据库](http://en.wikipedia.org/wiki/Microsoft_Access)   
　　Access数据库也是文件型的数据库，支持的很多SQL特性都类似于SQLite。自从Windows 2000开始，Windows就内置了Access的数据库引擎（[Microsoft Jet Database Engine](http://en.wikipedia.org/wiki/Microsoft_Jet_Database_Engine" \t "_blank) ）。所以Access数据库在上述系统中也是可以独立运行的（不依赖Office）。  
　　Access数据库最主要的缺点就是不能跨平台。另外还有几个小缺点：文件大小有限制（2GB）、不支持内存数据库。  
  
　　◇其它文件型数据库  
　　其实，除了Access之外，还有另外一些文件型数据库。但是这些文件型数据库要么名气太小，要么不支持多种编程语言（比如[HSQLDB](http://en.wikipedia.org/wiki/HSQLDB" \t "_blank) ），要么已经过时（比如FoxPro、Paradox）。所以后面分析应用场景的时候就不再提及这些玩意儿。  
  
　　◇CSV文件  
　　CSV（Comma Separated Values，详细解释见“[这里](http://en.wikipedia.org/wiki/Comma-separated_values) ”）是一种很简单的纯文本格式。它本身就是用来表示二维的数据信息的。一个CSV文件可以理解为数据库的一张表。  
　　CSV的缺点主要在于：不便于存储非文本的数据信息（比如BLOB类型的信息）；如果需要同时存储多张表的信息，就需要对应有多个CSV文件（文件一多，就嫌麻烦）。  
  
　　◇XML文件  
　　XML文件想必大伙儿都知道，我就不多说了。XML格式主要缺点也有两个：一个是由于XML本身是树状结构，有时候不便于表示二维数据表的信息；另一个是数据量大（比如文件超过10MB或者XML节点层次很深）的时候，解析XML的开销蛮大的。  
  
　　★**作为数据库的应用场景**   
　　前面说了一大通，现在开始切入正题，先说说SQLite作为一个轻型数据库，方便干哪些事儿？  
　　在这类场景中，由于是把SQLite作为数据库来使的，所以基本不用考虑CSV和XML这两种替代方案。  
  
　　◇需要数据库的小型桌面软件  
　　如果你开发一个小型的桌面软件并且需要用到数据库功能（比如某个背单词软件），那SQLite是一个不错的选择。因为SQLite很绿色又很短小精悍。  
　　不过，由于Windows在桌面系统的比重很大。对于那些**不考虑** 跨平台的开发人员，SQLite相对于Access来说，没有太大的优势。  
  
　　◇需要数据库的手机软件  
　 　眼下手机应用的发展很迅猛，相应的开发人员也多起来了。假如你就是一个手机应用程序的开发人员，并且你开发的应用需要有数据库功能（比如某个字典工具），那SQLite简直是不二之选。由于手机操作系统名目繁多，同时手机的内存偏小，这时候SQLite跨平台和轻量级的特长就充分发挥出来了。目前几个知名的手机操作系统（比如[Android](http://en.wikipedia.org/wiki/Google_Android" \t "_blank) 、[Windows Mobile](http://en.wikipedia.org/wiki/Windows_Mobile" \t "_blank) 、[Symbin](http://en.wikipedia.org/wiki/Symbian_OS" \t "_blank) 、[Palm](http://en.wikipedia.org/wiki/Palm_%28PDA%29" \t "_blank) 等），SQLite都支持得不错。  
　　在这种场合，Access基本没戏。  
  
　　★**作为数据容器的应用场景**   
　　所谓数据容器，就是把SQLite作为装数据的容器，充分发挥SQLite单一数据文件的优点。另外，还可以避免自己定义一套数据文件格式的麻烦。要知道，定义一个**完善的** 数据文件格式是难度极大滴（要考虑可扩展性、要考虑向下兼容、假如跨CPU架构还要考虑字节序、假如数据量大还要考虑性能、还要...）。  
  
　　◇数据备份/恢复、数据导入/导出  
　　某些软件系统（尤其是些企业应用系统）经常会碰到数据备份/恢复的功能需求。比如说，客户会要求你把一些数据（往往是业务相关的）定期备份成一个**独立的** 数据文件，然后存储在别处。一旦软件系统自身发生不测，再把备份的数据恢复回来。  
　　另外，导入/导出功能也是经常碰到的。一般是某个软件安装在多个地方。然后需要把一些数据（往往是业务相关的）从A处导出，然后在B处导入。  
　　针对上述这两种需求：如果牵涉的数据比较大，就不宜使用XML或者Access；如果牵涉到跨平台，就**无法** 使用Access；如果牵涉到多种数据，就不宜使用CSV（除非你能忍受多个CSV文件并存）。有上述条件限制的地方，都很适合用SQLite。  
  
　　◇在线升级  
　　这年头不联网的单机已经很少了，提供在线升级功能的软件会多起来。一般的在线升级分为两类：升级程序（比如Firefox自动升级新版本）和升级业务数据（比如杀毒软件升级病毒库）。这两种类型都可以使用SQLite来完成。把需要要升级的内容放置到SQLite数据库文件中，将来升级时只需下载**单一** 的升级文件即可。  
　　在这种场景，不太合适用CSV和XML。如果不考虑跨平台，倒也可以用Access来搞定。  
  
　　★**作为内存数据库的应用场景**   
　　在这种类型的场景中，咱们要充分发挥SQLite内存数据库的特长。由于SQLite的API设计比较合理，操作内存数据库和操作文件数据库几乎没啥区别，所以从文件型切换到内存型，代码不用大改。另外，从3.6.11开始，SQLite增加了[online backup](http://www.sqlite.org/backup.html" \t "_blank) 接口，便于在内存数据库和文件数据库之间进行数据的同步。  
  
　　◇降低磁盘I/O开销  
　　比如开发了某个字典工具，词库存储在SQLite数据库文件中。当词库越来越大的时候，你可能会发现，查词的速度越来越慢。当然啦，速度慢未必是磁盘 I/O引起的。这时候你可以把程序略微修改一下（可能就10行左右的代码），在初始化时把词库载入内存的SQLite数据库中。然后再对比测试一下性能。如果发现性能有明显提升，那你以后就可以一直用这种方式了。  
　　使用这个招数，要小心内存数据库对内存空间的占用。比如对于普通的PC，占用个几兆、十几兆还行，再大的话就不爽了。另外，对于手机操作系统，此招数效果不好（手机本身的内存就不是很大，而且存储介质的速度已经蛮快了）。  
  
　　◇作为临时表  
　　内存数据库方式，还可以用来充当临时表，存放一些临时数据。当程序的进程退出时，内存数据库就自然消失了，不会留下任何垃圾。  
　　不过这种方式只适合于某个程序独占临时表的情形。如果临时表需要被多个进程共用，这招就不灵了。

**版权声明**   
本博客所有的原创文章，作者皆保留版权。转载必须包含本声明，保持本文完整，并以超链接形式注明作者[编程随想](mailto:program.think@gmail.com) 和本文原始地址：

<http://program-think.blogspot.com/2009/04/how-to-use-sqlite.html>