适用于嵌入式设备的数据库查询技术①

林 鸿[1], 蔡坚勇[1,2,3,4]

[1](福建师范大学 光电与信息工程学院, 福州 350007)
[2](福建师范大学 医学光电科学与技术教育部重点实验室, 福州 350007)
[3](福建师范大学 福建省光子技术重点实验室, 福州 350007)
[4](福建师范大学 智能光电系统工程研究中心, 福州 350007)

摘 要: 本文以加快嵌入式数据库 SQLite 的数据查询速度为出发点, 提出了一种在原有 SQLite 的 B+树索引机制的基础上改进的新索引机制, 在对嵌入式内存资源影响不大的前提下提高了 SQLite 的查询速度.

关键词: 嵌入式数据库; SQLite; B+树索引机制; 内存资源; 查询速度

Database Query Techniques Suitable for Embedded Devices

LIN Hong[1], CAI Jian-Yong[1,2,3,4]

[1](College of Photonic and Electronic Engineering, Fujian Normal University, Fuzhou 350007, China)
[2](Key Laboratory of Optoelectronic Science and Technology for Medicine Ministry of Education, Fujian Normal University, Fuzhou 350007, China)
[3](Fujian Provincial Key Laboratory for Photonics Technology, Fujian Normal University, Fuzhou 350007, China)
[4](Intelligent Optoelectronic Systems Research Centre, Fujian Normal University, Fuzhou 350007, China)

Abstract: In order to accelerate the query speed of embedded database SQLite, we propose an improved index mechanism based on the original B+ tree, which can actually improve the query speed with little influence on the embedded memory resources.

Key words: embedded database; SQLite; B+ tree; memory resources; query speed

现如今, 随着嵌入式数据库技术的不断发展, 嵌入式数据库已经广泛应用于嵌入式领域, 人们对嵌入式数据库性能的要求也在不断地提高. 衡量嵌入式数据库性能的一个重要指标是嵌入式数据库查询机制的好坏. 而其查询操作的性能在很大程度上取决于嵌入式数据库所采用的索引机制. 由于嵌入式设备存在内存小、磁盘空间不足等缺点, 设计一种既能实现高效查询又不浪费内存和外存空间的索引机制显得尤为重要[1].

目前大部分嵌入式数据库都采用 B+树数据结构作为其索引机制. B+树具有查询效率高的特点, 不仅能够动态调整平衡, 还能同时支持关键字的随机查找和顺序查找. 但由于 B+树索引文件存在于外存中, 在进行查找时要进行一定次数的 I/O 操作, 这个在一定程度上势必会影响到数据库的查询速度[1-2]. 本文采用一种新的索引机制来解决这个问题, 即在嵌入式数据库 SQLite 原有索引机制的基础上, 将 B+树索引文件中的读取过的文件指针以 T 树的形式存放在内存中, 从而在对内存资源影响不大的情况下提高了 SQLite 的数据查询速度.

1 基本原理介绍

1.1 SQLite 索引机制的发展

早期的 SQLite 采用的是 GDBM 的数据结构, 这种数据结构采用 Key-Value 的方式来存储数据, 其特点是高效的查询, 低效的插入, 只适合存储比较静态的数据. 在中期, SQLite 则采用 B-树索引机制, 这种方法查询效率高, 并且确保了一定的空间利用率. 但由于 B-树的非终端结点中包含有指向实际数据的指针, 结点的结构比较大, 结点可能要分多次读取到内存中,

① 收稿时间:2013-09-14;收到修改稿时间:2013-11-11

这样造成的多次 I/O 操作势必会影响到检索数据的速度[2]. 而现在的 SQLite 采用的是 B+树数据结构, 它是一种树型结构, 是对 B-树数据结构的改进. B+树的检索速度比 B-树快[4], 但是比 B-树占用更多的空间资源.

1.2 B+树索引机制[3]

1.2.1 B+树查找算法

B+树有两种查找方式: 一种是从根节点开始进行缩小范围的查找; 还有一种是从叶子结点从小到大进行顺序查找. B+树的缩小范围的查找和 B-树的查找方式是类似的. 不同的是, 在进行缩小范围的查找时, 不管成功与否, 都必须查找到叶子结点才能结束. 在结点内查找时, 若给定的关键字 K 小于或等于结点内的关键字 K_i, 那么应该继续在 P_i 所指向的子树中进行查找. 具体算法如图 1 所示.

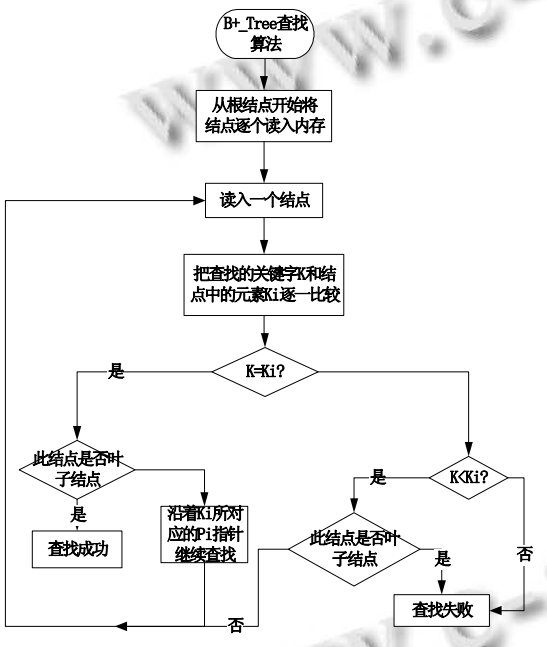

图 1 B+树查找算法

值得一提的是, 数据库索引本身也比较大, 人们通常将索引以文件的形式组织起来存放在外存中. 所以, 在查找过程中, 无论是 B-树还是 B+树, 它们从索引文件中将每一个结点读入内存的过程都需要进行一定次数的 I/O 操作, 并且进行 I/O 操作的次数取决于结点的大小和磁盘中一个盘块所能容纳的大小. 由于外存和内存的存储介质不同, 与内存的存取相比, 外存上的存取所花费的时间往往要高好几个数量级. 所以, 减少 I/O 操作次数是提升嵌入式数据库查询速度的关键因素.

1.2.2 B+树的优缺点

(1) B+树较 B-树具有更低的磁盘读写代价

在 B+树中, 非终端结点中只包含有关键字以及指向子树的指针, 并没有包含指向关键字具体信息的指针, 也就是说, 从结点大小来看, B+树的结点要比 B-树来的小. 这样, 从索引文件中读取结点所需要的 I/O 次数(即磁盘的读写代价)就相对来的少. I/O 次数的减少从根本上提升了索引文件读写的速度. 这是 B+树比 B-树更优的地方.

(2) B+树具有更稳定的查询效率

对于 B+树的查询机制来说, 无论是否查找到关键字, 都需经历一条根到叶子结点的道路.

(3) B+树的空间利用率低

B-树的任何一个关键字有且只有出现在一个结点中, 而 B+树的关键字出现在多个结点中, 所以相比之下, B-树有更高的空间利用率.

1.3 T 树索引机制[4]

T 树是一棵特殊的平衡二叉树, 它与普通的平衡二叉树的区别在于, 平衡二叉树的每个节点中只存放了一个关键字信息, 而 T 树的一个结点中存放了多个并按照键值排序后的关键字信息, 这样从一定程度上保证了结点的空间占有率. (结点结构如图 2 所示)

T 树的特点:

A, T 树由三个指针(父结点指针、左结点指针、右节点指针)、一个有序数组(存放数据指针)和一些控制信息组成;

B, T 树的平衡因子的绝对值不大于 1;

C, T 树结点中最左边和最右边的键值分别存放的是最小和最大键值, 并且键值按照从小到大的顺序排列. 其左子树存放的是比它的最小键值小的关键字, 其右子树存放的是比它的最大键值大的关键字.

由于 T 树的有序数组中存放的是指向数据的指针而不存放数据, 这样内存的消耗就比较小, 从而减少了内存的负担. 因此, T 树比较适用于内存数据库.

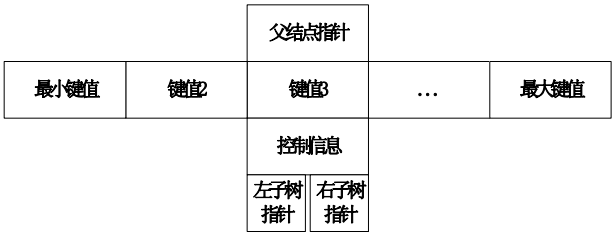

图 2 T 树结点结构

从以上的原理分析可以看出，B+树比较适用于文件系统的读写，T 树比较适用于内存数据库的读写，而减少 B+树的 I/O 操作次数是提升嵌入式数据库索引文件的查询速度的关键因素.

2　SQLite索引机制的改进方案设计

2.1 改进的 SQLite 数据库采用的索引机制方案

为了减少 SQLite 的索引机制的 I/O 次数，本文考虑将索引文件中读写过的数据的文件指针存放在内存中，以便下一次快速调用. 为了便于管理，将文件指针以更适合于内存的 T 树的数据结构组织起来并存储在内存中. 由于嵌入式设备的内存有限，将读写过的数据的文件指针存放在内存中势必会占用一定的内存空间. 故考虑给 T 树中存放的文件指针数设置一个阈值，当文件指针数超过此阈值时，就将长期未使用过的文件指针删除，这样做可以保证此方案不会消耗过多的内存资源.

存放文件指针的 T 树结点结构如下:

```
typedef  struct  TTreeNode
{
  short  sBalance;          /*平衡因子*/
  unsigned short usItems;/*实际元素数目*/
  struct TTreeNode* left;   /*左子树结点*/
  struct TTreeNode* right; /*右子树结点*/
  long  lItem[MAXSIZE];   /*关键字*/
}TTreeNode;
```

2.2 改进机制数学分析

设 N_{im} 为从内存中读取文件指针的次数，T_{im} 为内存中读取文件指针的平均时间，从索引文件中读取文件指针的平均时间为 T_{fm}，则节省的时间为 $T=(T_{fm}-T_{im})*N_{im}$

设 M_C 改进后增加的内存消耗，$MaxSize$ 为 T 树结点中的关键字个数，故 T 树的结点占有的字节数为 $Num=(MaxSize*4+12)Bytes$

若设 N_T 为读取个数，则由于 T 树所引起的内存消耗值为: $M_C=Num*N_T/Maxsize$

如果 $MaxSize$ 是个定制，那么由 T 树引起的内存消耗会随着 N_T 的增大而增大. 由于先前设定的最大文件指针数为 α，所以 T 树内存增加范围在 $[0,(MaxSize*4+12)*a/MaxSize]$ (单位为 Bytes) 之间.

2.3 改进机制的查询数据算法

改进后的机制查询数据算法(如图 3 所示):

(1) 在存放着读取过的数据文件指针的 T 树内判断要查询的数据的文件指针是否存在于内存中.

(2) 若存在即命中，则在磁盘的数据库文件中将文件指针所对应的数据读取出来.

(3) 若没有命中，则按照原始数据库的 B+树索引机制来对数据进行查找.

(4) 若查找成功，则将此数据的文件指针写入内存的 T 树中，如果此时 T 树的文件指针数已经超过阈值，则将最不经常使用的数据删除后再将新数据插入.

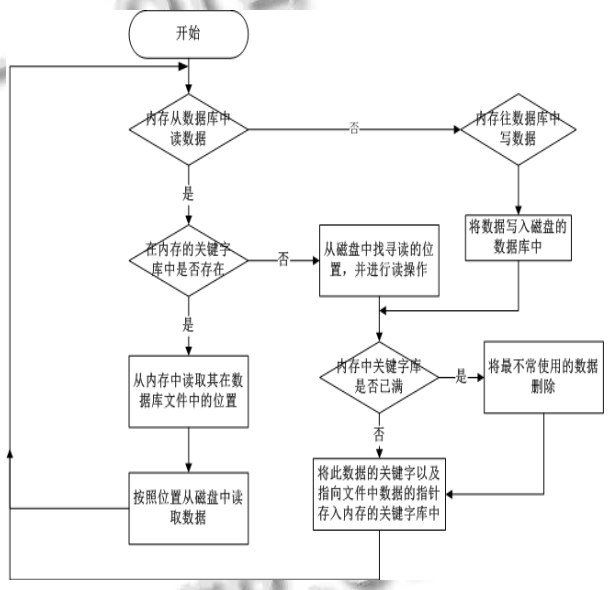

图 3　改进后的查询算法

通过将从索引文件中读取出来的数据文件指针以 T 树的结构保存在内存中减少读取索引文件的 I/O 操作次数，从而改进索引机制达到增强嵌入式数据库查询性能的目的.

3　索引机制的实现与测试

本文使用 C++模拟开发了 SQLite 的 B+树索引机制模块，按照上面介绍的方案对其进行改进，并将改进前和改进后的索引读取时间和内存消耗进行了对比测试.

测试平台: Mini2440 ARM-Linux

测试步骤:

(1) 由于B+树数据结构的阶数M对索引数据的读取速度有一定的影响，所以必须先将改进前和改进后

的阶数 M 设置为相同的值.

(2) 往索引文件“BPlusTreeData.dat”按顺序插入 500 条索引数据, 设置 Maxsize 为 3.

(3) 分别对拥有 500 条索引数据的索引文件分别进行 100、200、300、400、500 次的随机查询, 并比较改进前和改进后的所花费的时间.

(4) 对拥有 500 条索引数据的文件进行 200、400、600、800、1000、3000、5000、7000、9000、10000 次的随机查询, 并比较改进前和改进后的内存消值.

测试结果:

如图 4 所示, 在查询速度方面, 经过一定次数的查询, 改进后的查询速度明显比改进前的有所增加.

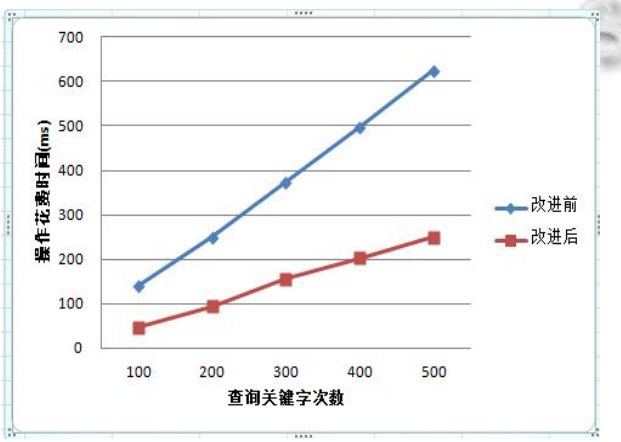

图 4 改进前与改进后花费时间比较

从测试结果可以看出, 改进后的机制操作所花费的时间比起改进前有了明显减少, 其中分别经过 100~500 次关键字查询节省的时间分别为:

$$T_{100}=(T_{fm}-T_{im})*N_{im}=145-47=98ms$$
$$T_{200}=(T_{fm}-T_{im})*N_{im}=252-94=158ms$$
$$T_{300}=(T_{fm}-T_{im})*N_{im}=375-156=219ms$$
$$T_{400}=(T_{fm}-T_{im})*N_{im}=496-202=294ms$$
$$T_{500}=(T_{fm}-T_{im})*N_{im}=625-250=375ms$$

在内存消耗方面, 经过测试发现, 改进前的内存消耗值恒为 948K, 改进后的内存消耗值随着 NT 值的增加而增加, 但是当 500 条索引数据都插入 T 树后, 内存消耗就达到一个定值. 内存增量(内存增量为改进后与改进前的内存消耗差值)如图 5 所示, 当查询次数达到 9000 次以后, 内存增量就达到一个定值, 不会再随着查询次数的增加而增加, 此时内存增值也只有 240K, 对内存资源影响不大, 所以只要严格控制好 α 的值, 此方案就不会带来太多的内存消耗.

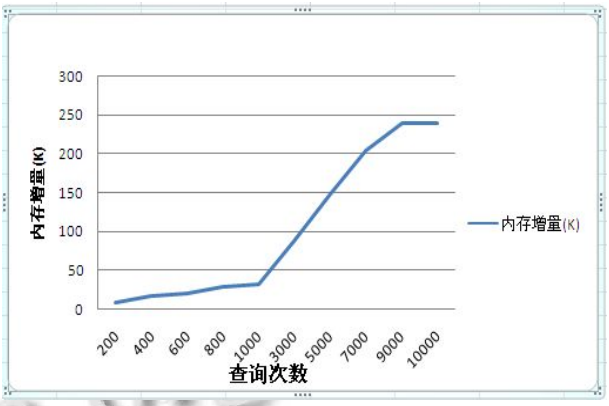

图 5 内存增量图

4 结语

本文在嵌入式数据库 SQLite 原有的 B+树索引机制的基础上提出了一种新的索引机制, 即将从 B+树索引文件中读取过的数据指针以 T 树的数据结构存储在内存中, 从而在对内存资源影响不大的情况下, 减少了对索引文件的 I/O 次数, 提高了 SQLite 的数据查询速度.

参考文献

1 邵慧琳.嵌入式数据库索引机制研究[硕士学位论文].重庆:重庆邮电大学,2011:18–34.

2 颜丽,王伟,童治军.嵌入式数据库 SQLite 中 B 树的研究,九江职业技术学院学报,2012,4.

3 宋玲,杨雪君,马兰.嵌入式内存数据库的存储和索引算法研究.计算机科学与探索,2010,4(8).

4 文继军,王珊,张文亮.嵌入式数据库中 B+树索引的设计和实现.计算机科学,2003.

5 王晓,陈永春.嵌入式数据库关键技术及其发展趋势.哈尔滨师范大学自然科学学报,2012,(28):67–69.

6 张学琴.嵌入式数据库 B+_树索引机制研究及改进.计算机与现代化,2009,(12):68–71.

7 夏铭.嵌入式数据库结构及索引查询技术研究[硕士学位论文].安徽:合肥工业大学,2007:39–44.

8 黄加喜,陈天煌,郑胜英.嵌入式数据库索引机制的研究.计算机安全,2008,9:54–57.

9 陈燕红.嵌入式数据库查询处理研究[硕士学位论文].新疆农业大学,2012.