

数据库快速查询方法研究与应用

李芳芳, 田志军

(云南师范大学 商学院, 云南 昆明 650106)

摘 要: 提出了一种优化大型数据库性能的方法. 通过调整内存参数优化, 设计索引的优化这两方面来对大型数据库系统进行优化, 以求使数据库的性能得到优化. 实验结果表明, 通过本文方法优化后的数据库在运行时间, 搜索速度等性能上都有了很大的提高, 因此得出, 本文的方法在改进数据库性能方面, 有很好的效果.

关键词: 数据库查询优化; 内存参数优化; 设计索引

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2012)03-0163-04

Rapid Database Search Method of Research and Application

LI Fang-fang, TIAN Zhi-jun

(Yunnan Normal University Business School, Kunming 650106, China)

Abstract: Presents a large-scale database performance optimization methods. set parameters by adjusting the design optimization and set index to optimized large database systems. Experimental results show that the method optimized by the database at run time, the search speed performance has greatly improved, thus obtained, this approach to improve database performance, have a good effect.

Key words: database search optimization; parameters optimization; set index

1 引言

随着企业的业务系统的规模的扩大, 业务数据不断膨胀, 各种大型的数据库在企业中的作用十分重要^[1-2], 数据库查询性能起着至关重要的作用. 近年来, 针对数据库查询性能如何提高的问题, 提出了许多如何提高数据库查询性能的方法和理论. 但是当前常规的方法大都是靠增加硬件设备, 或者提高硬件的档次来对数据库进行性能上的调整, 这样的优化措施不但增加了企业的成本, 有时开销还是非常的大. 然而, 只是从硬件的角度来对数据库进行性能方面的调整, 还是远远不够的^[3-4].

基于以上的问题, 本文提出一种新的数据库查询优化方法, 主要是从软件的角度来对数据库进行性能优化和调整. 本文的方法不需要额外增加企业的负担, 在企业原有的硬件设施的基础上完成对企业所使用的数据库的优化工作, 有着较强的实用性和经济效益^[5-6].

2 数据库查询优化的标准

对于普通的数据集合, 主要从以下两个方面测试数据库的查询性能:

(1) 程序吞吐量: 吞吐量是程序在应用过程中完成任务量与完成任务需要时间的比值. 这个比值客观的反应了系统工作能力. 下面用表达式阐述一下吞吐量的含义:

吞吐量 = 完成的任务 / 完成任务消耗时间

(2) 数据集合的程序在收到使用者指令后做出处理需要的时间: 一般情况下, 这里的时间是指使用者对程序做出指令或者使用功能时开始计时, 程序针对该指令做出处理时结束计时. 从开始计时到结束计时的这段时间就是数据集合的程序在收到使用者的指令后做出处理需要的时间. 这个时间与程序的吞吐量有一定的关系, 假设吞吐量增多, 则这个过程需要的时间会变少.

以上 2 个标准作为数据库系统优化的一个指标

量,也是优化的目的.本文将结合当前市场上比较常见的 DB2 数据库系统,测试本文用到的数据库调优方法^[3].

3 数据库的查询优化方法

3.1 调整实例内存参数提高数据库性能

通常在安装了数据集合相关程序后,其相关参数都是默认的.大部分的用户都不会去调整这些参数,这就造成了数据库性能的降低.可以通过对数据库内存改变相关系数,从而增强数据集合使用过程中程序处理能力,提高了数据集合性能.

针对 DB2 数据集合,有三种相关系数,包括实践中使用的系数,数据集合中使用的系数,DB2 注册分量使用的系数.这三种系数都是数据集合使用的基础.在改变相关系数前,首先确定 DB2 数据集合构造方式.在这个数据集合中,包含的所有数据子集都使用实践过程中的方式进行管理.一种实践过程中的方式可以用来管理数量很大的数据子集.所以,数据集合构造方式分为实践过程中可以分享的内存和数据集合中可以分享的内存.

每次实践过程中使用的数据集合,都是实践过程中可以分享的内存.这种内存是在数据集合管理器开始时候得到的,并且在数据集合管理器结束使用时,将不再能继续使用.这种内存集合常在实践过程中使用,比如监视、审核等时候.这个内存也是可以改善性能的一个重要区域.调整内存里数据集合的缓冲区域,日志的缓冲区域,能够提高数据系统的性能.以下是数据集合管理器系数控制实践过程中内存的分享方式和缓冲区域的限制:

- (1) 实践过程中的内存.
- (2) 检测系统:主要作用是检测对象.

缓冲区域系数是在实践管理的过程中提前留出的内存空间.用来暂时存储实践过程中的相关数据,这样可以避免频繁的将数据存储到磁盘中,降低数据集合的性能.数据集合依据检测系统和审核缓冲的空间,经过相关运算获取在当时的条件下需要的内存空间.

假设缓冲区域是一个具体的值,则实践过程中需要的内存空间就必须保证在某一个具体的范围里.这样才能保证数据集合相关程序的正常运行.如果超过这个范围,内存区域的使用就会出现错误.依据数据集合的实际使用情况,修改这些内存参数,可以大大的提高数据库的性能,避免了内存的过度浪费.

3.2 调整共享内存参数提高数据库性能

所有的数据集合都有各自分享的内存空间.它是数据集合首次使用的时候,系统自动给予的,而这个分享内存空间的大小也是系统自动给予的.这个内存空间在数据集合没有使用过的时候是不存在的.这个内存空间一般在数据集合执行任务的时候使用,可以进行数据的备份、恢复等操作.它直接关系到数据集合的工作能力.图 1 阐述的是内存构造方式.

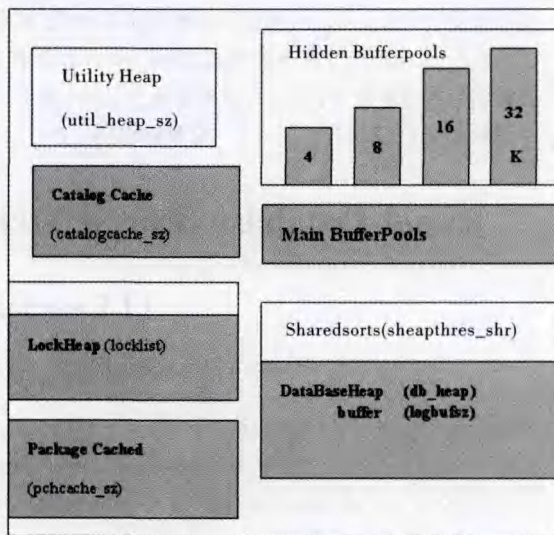


图 1 DB2 数据库共享内存结构

在上面的图表中绿色标识区域代表数据集合初次使用时系统默认的内存空间以及内存空间分配情况.

在上面的图表中主要缓冲区域是数据集合分享的内存空间中最大部分.在数据集合内存空间中,控制全部的普通数据和索引数据.是内存空间中最重要的一部分.每一个表空间的建立,都需要拥有与其对应的缓冲区域.所以数据集合中默认的系数可以改成较大的数值,这样会增加缓冲区域的范围,导致数据集合的工作能力得到一定程度的提高.但是并非缓冲区域越大数据集合的工作能力越强.假设缓冲区域达到极大值,那么也许会导致缓冲区域与可以找到的内存空间的地址不匹配.在这种情况下,数据集合将无法使用,所以一个数据集合至少有一个缓冲区域.下面阐述一下合理分配缓冲区域的方式.在数据集合中使用缓冲区域命中率判断.缓冲区域命中率代表数据集合管理器可以自动分配时间.缓冲区域命中率与磁盘使用的频率成反比.下面的表达式是数据集合中缓冲区域命中率的运算方式:

$$\text{BPHR}(\%) = (1 - ((\text{"Buffer Pool data phys-}$$

ical reads”+“Buffer pool index physical reads”)/ (“Buffer pool data logical reads”+“Buffer pool index logical reads”))) * 100

上面提及系数的取值可以利用数据集合缓冲区域快照得到. 假设经过运算获取的缓冲区域取值小于 90%, 那么缓冲区域可以变大, 经常使用的大表所用的内存区域可以得到一个缓冲区域, 经常使用的一组小表也可以得到一个缓冲区域. 如果在最佳的状态下, 此命中率一般高于 96%, 甚至能够接近 100%.

改变缓冲区域和日志缓冲区域都能够优化数据集合的工作能力. 日志缓冲区域的范围取决于数据集合的系数 LOGBUFSZ. 此系数代表可以被用于将日志存入磁盘前的缓冲区域取值. 假设系数过小, 则在数据集合使用过程中需要不断将日志存入磁盘, 这样会影响数据集合的工作能力. 在数据集合中利用快照判断 LOGBUFSZ 系数是否是最合适的, 利用 get snapshot for all on dbname 指令得到两个系数的取值:

Log pages read=0

Log pages written=12644

在系统运行的时候完成任务, Log pages read 取值相对比较小, 从磁盘读取日志的频率较低. 一般系统默认这个取值是 256 页. 通常“Log pages read”和“Log pages written”的比值也相对比较小. 在最佳状态下, “Log pages read”的取值应该是 0, 而“Log pages written”的取值相对比较大. 假设 log pages read 过多, 则代表需要相对比较大的 LOGBUFSZ.

由此可见, 根据企业自身的情况, 随时调整这些数据库中内存的参数, 对数据库性能的优化有着很重要的作用, 可以让数据库的性能大大的提高.

4 合理设计索引提高数据库查询性能

在一个工作程序中, 索引也处于关键的地位, 合理的建立索引也可以提高数据集合的性能. 索引是有序序列的数据子集的排列. 通过索引可以方便准确的获得需要使用的信息在数据集合中的具体位置, 从而提高了获取信息的效率, 并且索引是基于关系的数据集合中不可替代的条件方式. 合理的根据实际情况建立索引提高了数据集合的相关性能, 尤其是查询方面的相关能力. 在 DB2 数据集合里是利用 B+ 树保存索引的. 在 DB2 数据集合和 oracle 数据集合内都包括集中形式构造的索引和不集中形式

构造的索引. 如图 2 所示.

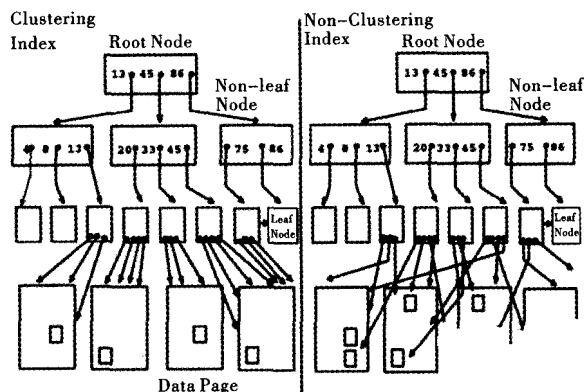


图2 集群索引与非集群索引

通过上面的解析图可以得知, 集中形式构造的索引在进行数据搜索的时候可以搜索到具体的页数, 然后根据页数提取数据集合中的相关数据. 而集中形式构造的索引在数据搜索的时候搜索到相关的页数, 根据页数在数据集合中获取相关的内容. 因此在建立一个数据集合索引的时候, 可以建立一个集中形式构造的索引和多个不集中形式构造的索引.

4.1 创建集群索引提高查询性能

利用数据集合中的某一个有序的序列构造集中形式的索引, DB2 数据集合管理器根据序列的内容储存其相关的页数, 集中形式的索引可以在数据查找的时候方便快捷的获取数据所在的页数, 也可以在数据集合中有新的数据的时候, DBA 内部存储核心将新加入的数据存储在相同种类数据的周围. 这样根据相同的属性可以快速便捷的查找新插入的数据. 比如, 一个基于售出时间的集中形式索引, 假设待检测的时间的有关内容就在周围, 则需要较少的 I/O 就可以获取相关内容, 提高了系统的性能. 因此针对一个数据集合建立集中形式的索引可以在很大程度上提高系统的性能, 但是在插入相关数据的时候, 内部存储核心会利用集中形式的索引确定插入的数据的合适的位置, 从而降低了插入或者更新行为的可操作性.

4.2 创建非集群索引提高查询性能

一个数据集合可以对应多个不集中形式的索引, 但是索引的数量与系统的性能并不一定成正比. 也就是说, 并非是索引数量越多, 系统的相关性能越高. 不集中形式的索引是对数据集合中有序序列的内容进行复制, 在创建了不集中形式的索引后, 有序序列则可以同时包括数据相关的存数页数和索引页数, 根据不集中形式的索引可以获取需要查找的数

据的相关内容和索引的相关内容. 因此需要很大的存储空间. 在插入新的数据的时候, 也需要在不集中形式的索引上插入相关的内容. 假设对数据集合的有序序列进行更新, 那么需要同时获取更新的内容的数据页和索引页. 因此如果不集中形式的索引超过了合理的个数, 则会降低插入或者更新数据的可操作性. 下面阐述建立不集中形式索引的条件:

(1) 针对数据集合内所有主要有序序列和大多数的附属有序序列创建不集中形式的索引. 大部分的查询都需要同时使用主要有序序列和附属有序序列, 所以要在最大限度内建立与其相关的不集中形式的索引, 从而优化系统的查询方式, 提高系统的性能.

(2) 假设数据子集出现在 WHERE 子句里的频率较高, 那么需要对其建立不集中形式的索引. 但是谓词相对较小的过滤时排除在外. 针对特殊情况下的查找, 可以依据数据集合中所有相关的有序序列构造一个联合的不集中形式的索引, 包括 SELECT 语句和 WHERE 子句中存在的有序序列. 联合的不集中形式的索引通常包括很多有序序列, 这些序列的排序是非常重要的, 此特点是依据索引的 B+ 树构造形式得知的. 通常在索引的排序过程中谓词的使用频率起决定性作用. 使用的频率越高, 在索引中的位置越靠前. 而联合的不集中形式索引则排在最后.

(3) 假设一个有序序列已经建立了不集中形式的索引, 则不应该使用函数. 下面陈述一下不能使用函数的理由: 假设在已经建立的不集中形式的索引上使用了函数, 考虑到其不稳定性, 则经过函数处理后获取的结果和存入的数据也许会不是一一对应的关系, 则导致在不集中形式索引中排序差距比较大的很多个有序序列的内容都符合有函数的谓词的相关条件, 索引 DB2 数据集合的优化程序不能将其匹配, 甚至会致使程序直接扫描有序序列的内容. 所以, 不能对使用函数的内容建立不集中形式的索引.

5 实验结果仿真

本文的算法在计算机上进行了仿真实验. 实验环境如下:

(1) 系统采用 Window XP 系统上运行的 DB2 Universal Database Workgroup9. 5.

(2) 客户端读本地数据; 8 个文件, 3 个小时、采样率为 12 数据, 共 129600 行, 数据类型 float, 每个文件大小为 517KB

(3) 数据库表格除信号外还包括运行时间, 仿真时间, 和相对时刻三个字段.

表 1 给出了没有运用本文算法和运用本文方法优化之后的数据库时间对比, 本文选取了 10 次的实验结果, 选取了平均执行时间, 作为参考的依据.

表 1 实验对比

	数据吞吐量	执行指令时间
常规数据库	85 KB/s	1.24 s
优化后的数据库	117 KB/s	0.77 s

可以看出本文方案明显提高了性能, 每 10 次为一个测试组但愿, 吞吐量性能提高了约 38%. 在实验中, 取十条常规指令执行作为测试的基本单元. 可以看出, 优化后的数据库性能提高了约 38%. 由此得出, 本文的数据库优化方法, 实用有效.

6 结束语

本文在传统的数据库应用的基础上, 加入了根据数据库实际使用情况, 调节内存参数和索引的软优化方法, 利用该方法不会增加企业的额外成本, 优化的效果也好, 还避免了大规模更换硬件设备的麻烦. 因此, 本方法在数据库应用领域有着非常重要的实际应用价值.

参考文献:

- [1] 牛新庄. DB2 数据库性能调整与优化 [M]. 北京: 清华大学出版社, 2009.
- [2] 肖军. ORACLE 数据库性能调整与优化 [D]. 武汉: 武汉大学, 2004.
- [3] 朱延梅, 邵远山. 基于 DB2 数据库应用系统的性能优化方法 [M]. 合肥: 合肥管理学院, 2006.
- [4] 陶剑文. 一种分布式 Web 日志挖掘系统的设计与实现 [J]. 计算机仿真, 2006(10): 133-134.
- [5] Krista Rizman, Zalik. An efficient K-means clustering algorithm [J]. Pattern Recognition Letters, 2008 (29): 1385-1391.
- [6] 贺金鑫, 徐高潮, 付晓东, 等. 通用数据库性能测评系统的设计与实现 [J]. 微电子学与计算机, 2009(01): 44-46.

作者简介:

李芳芳 女, (1979—), 硕士, 讲师. 研究方向为信息化与教学管理、数据库与信息系统、软件项目管理.

田志军 男, (1977—), 硕士, 讲师. 研究方向为信息化与教学管理、数据库与信息系统、软件项目管理.