企业级应用系统中的数据库性能优化策略

Database performance optimization strategy in the enterprise level application system

任玉辉　王成良　（重庆大学 计算机学院　重庆 400030）

摘　要： 随着企业的不断发展，企业级应用系统的信息量和用户数也大量增长，这使得应用系统的性能越来越依赖于数据库的性能。但在企业数据库应用系统的开发过程中，很多程序员仅关注系统功能的实现，在系统访问性能方面考虑较少，但事实上随着企业数据量的急剧膨胀，要从大数据量的数据表中快速查询出相关数据已变得非常困难。基于这一问题，本文就如何优化大型数据库的性能进行了一些探索，提出了优化数据库访问性能的若干策略，并结合某实际应用详细说明了提高数据库访问性能的过程，可以为系统设计和开发人员提供有益的借鉴。

关键词： 数据库 访问性能 优化策略 应用系统 索引

1　引言

当前，数据库应用系统日益普及，特别是企业级数据库应用系统，具有数据量大、并行访问需求高的特征。因此，对于企业级用户来说，现代应用所面临的主要问题之一就是如何快速和高效地进行数据处理，而这一问题往往是在设计之初被开发人员所忽略的问题。这样系统往往会随着数据量的增多而出现性能问题。数据库的性能优化就成为人们研究的一个重要课题。

数据库性能优化的基本原则就是通过尽可能少的磁盘访问获得所需要的数据。本文从数据库设计、索引设计、查询语句优化几个比较共性的方面分析了数据库性能优化的问题，提出了若干数据库性能优化的策略，并结合实际介绍了优化技术在某企业物流综合管理系统中的应用。

2　数据库设计的优化

数据库设计包括逻辑设计和物理设计。逻辑设计包括使用数据库组件为业务需求和数据建模，而无需考虑如何或在哪里物理存储这些数据；物理设计包括将逻辑映射到物理媒体上，使用可用的硬件和软件使用户能尽可能快地对数据库进行物理访问和维护[1]。

2.1　逻辑数据库设计

数据库逻辑设计的主要目的是产生一个 DBMS 可处理的数据模型和数据库模式，该模式必须满足数据库的完整性、一致性及运行等方面的用户需求。数据库逻辑设计要达到除去冗余数据、提高数据吞吐量、保证数据完整性、清楚地表达数据元素之间关系的目的。规范化是达到这样目的的理论指导。一般数据库的逻辑设计要遵循规范化的前三级标准[2]：

(1) 第一范式：任何多值属性（又称重复组）都被除去，表中所有行和列交叉处都只有一个值；

(2) 第二范式：每个非关键字字段必须依赖于主关键字，不能依赖于一个组合主键的一部分；

(3) 第三范式：在第二范式的基础上，数据表中如果不存在非关键字段对任一候选关键字段的传递函数依赖则符合第三范式。所谓传递函数依赖，指的是如果存在"$A \rightarrow B \rightarrow C$"的决定关系，则 C 传递函数依赖于 A。

在数据库逻辑设计中遵循规范化准则可以减少数据冗余，也就减少了用于存储数据的页。冗余数据的减少相应的提高了系统的查询性能。但是，在这里要提出的是规范化并不总能提高性能，因为规范化设计要得到列数最少的表，这样使得一些查询要通过复杂的连接才能实现，这将导致性能的下降[3]。因此要对

规范化进行必要的平衡，必要的时候要用非规范的形式来提高检索速度，以便最大幅度地提高系统的性能。例如：

（1）如果操作由于规范化的原因必须要通过 4 路或更多的连接才能实现，就可以考虑在表中加入重复的属性列；

（2）常用的计算字段（如总值、均值、百分比等）可以考虑存储到数据库表中。

总之，要以规范化设计为出发点，然后出于特定的原因有选择地非规范化某些表。

2.2 物理数据库设计

在大多数情况下，一旦选定了系统所使用的数据库管理技术，许多物理数据库设计的问题就已经被确定下来。在物理数据库设计中，为避免系统的性能的降低和对数据完整性造成影响，开发人员应遵循以下一些重要策略：

（1）为逻辑数据模型中的每个属性选择恰当的数据类型，以便使存储空间最小而性能最佳，被索引的列更是如此。比如能使用 samllint 就不要使用 integer，这样索引字段就可以更快地被读取，而且可以在一个数据页上放置更多的数据行，从而减少了 I/O 操作。

（2）将数据库分区。将一个大表拆分成更小的表，可以提高查询速度，更快地执行维护任务。实现分区操作时也可以不拆分表，而是将表物理地放置在分开的的磁盘驱动器上，并把相关的表放在与之分离的驱动器上。这样，由于执行涉及表之间连接的查询时，多个磁头同时读取数据，可以大大地提高查询速度。分区的方法有[4]：

① 硬件分区。RAID 设备允许数据在多个磁盘驱动器中带化，使更多读/写磁头同时读取数据从而达到提高访问速度的目的。

② 水平分区。按照数据行对表进行分割，将一个表分为多个表，每个表包含相同数目的列和较少的行。这种方法适用于那些数据行作为子集被逻辑工作组访问的实体表。

③ 垂直分区。把一个实体表按照属性列进行分割得到属性列较少的表。对于那些属性列之间的访问频率差异较大的实体表使用这种分割方法可以得到很好的效果。

（3）使用文件组放置数据。将具有相似结构的记录放在同一文件组中。在 sql server 2000 中默认情况下，索引创建在基表所在的文件组上，不过可以在其它的文件组上创建非聚集索引。在其它文件组上创建索引，可以使用各个文件组自带的控制器和不同的物理驱动器从而实现性能的提升。另外，也可以通过在 RAID 0、1 或 5 设备上创建单个文件来达到同样的效果。

（4）优化数据库日志。优化数据库日志可以从以下几个方面来考虑：

① 将日志创建在物理上单独的磁盘或 RAID 设备上。因为日志文件按照序列写入，所以使用单独的专用磁盘可以保持磁头在一个写入操作的位置。

② 将日志文件的初始设置值和增量百分比设置为合理大小，以防止当需要更多的日志空间时文件自动扩展。因为在日志扩展时，会创建一个新的虚拟日志文件，这时写入日志的操作要一直处于等待状态，如果日志扩展得过于频繁，性能会受到不良的影响。另外，如果文件增长幅度与写入日志的日志记录数相比太小，也会造成日志的不断扩展，所以还需要将日志文件的增量百分比设置成合理大小。

③ 手工收缩日志文件而不是允许 sql server 自动收缩文件。当一个系统正在进行日志处理时可能会造成数据页的移动和锁定而妨碍性能。

3 索引设计优化

提高查询速度最有效的方法就是优化索引。索引是建立在实体表上的一种数据组织，它可以提高访问表中一条或多条记录的查询效率，但是未经调优的索引可能会导致一些不良后果[5]：

- 索引是需要付出维护代价的，应防止建立后却从未使用。
- 为了返回一个单一记录而扫描整个文件。
- 由于存在错误的索引，导致多表的连接操作持续时间过久。

优化索引可以避免扫描整个表，减少因查询造成的 I/O 开销。一般说来建立索引要注意以下几点：

（1）在主键上建立索引，尤其当经常用它作为连接的时候；

（2）在查询经常用到的列上建立非聚簇索引；

（3）在频繁进行范围查询、排序、分组的列上建立

聚簇索引；

(4) 在经常用于连接而又未指定为外键的列上建立索引；

(5) 尽量使用较窄的索引，这样数据页每页上能因存放较多的索引行而减少 I/O 操作；

(6) 在查询中经常作为条件表达式并且不同值较多的列上建立索引而不同值较少的列上不要建立索引；

(7) 经常同时存取多列，且每列都含有重复值，可以考虑建立复合索引来覆盖一个或一组查询，并且把查询引用最频繁的列作为前导列；

(8) 当数据库表更新大量数据后，删除并重新建立索引来提高查询速度；

(9) 频繁进行删除、插入操作的表不要建立过多的索引；

(10) 当对一个表的 update 操作远远多于 select 操作时，不应创建索引。

总之，建立索引一定要慎重，对每个索引建立的必要性都要仔细分析，一定要有建立的依据。过多的索引或不充分、不正确的索引对提升数据库的性能毫无益处。

4 查询语句的优化

建立必要的索引并不意味数据库的性能已经提升到了令人满意的地步，还要针对某些 sql 语句进行分析和优化。一个好的查询表达式不是基于纯粹的理论假设想象出来的，而是在实际的项目开发过程中总结的经验[3]。下面就通过几个常用的查询优化方法来说明 sql 查询语句优化技术。

(1) 字段提取按照"需多少，提多少"的原则，避免"select * "。"select * "需要数据库返回相应表的所有列信息，这对于一个列较多的表无疑是一项费时的操作。

(2) 避免使用 or，用 union 来代替。Or 语句的执行原理并不是利用列上的索引根据每个 or 语句分别查找再将结果求并集，而是使用了"or 策略"，即先取出满足每个 or 子句的行，存入临时数据库的工作表中，再建立唯一索引以去掉重复行，最后从这个临时表中计算结果。这样使用 or 可能造成索引失效，导致顺序扫描整个表，大大降低查询效率。

(3) 严格的选择条件写在前面，较弱的选择条件写在后面，这样就可以先根据较严格的条件得出数据量较小的信息，再在这些信息中根据后面较弱的条件得到满足条件的信息。

(4) 子查询展平。子查询展平是把子查询转换成联结来实现。对于主查询的每一条记录子查询都要执行一次，嵌套的层次越多效率越低。

(5) 避免对 where 子句使用数学运算符。即不要对数据表的属性列进行操作。Where 子句中对列的任何操作结果都是在 SQL 运行时逐列计算得到的，因此就不得不进行全表扫描而不能使用列上的索引。(6) 避免在 where 子句中使用非聚合表达式。因为非聚合表达式常常使索引失效而导致扫描整个表。

虽然一个查询可以由不同的查询语句来实现，但是消耗的时间却是不一样的。对 sql 语句的优化可以提高数据库检索的性能，具体的查询语句书写由具体情况来决定。

5 优化策略在某企业物流综合管理系统中的应用

某企业为全国各地的汽车生产商提供零部件生产服务，每天要完成对产品规格信息、产品出入库信息、产品订单信息和出货信息等大量信息的记录，该企业物流综合管理系统就是为了提高企业的工作效率加快信息化管理进程而设计实现的。

系统采用了 sql server 2000 数据库。系统的开发设计以各种型号的汽车零部件的信息分类为基础，通过系统实现产品分类、出入库、订单处理和生成发货单等工作的自动化、标准化、规范化。该系统采用的系统环境：计算机 CPU P4，内存 512M.；操作系统 windows 2000 Professional。

5.1 技术实施

本系统在使用之初，运行正常，性能良好，但是随着数据量的大量增加，访问性能问题就开始暴露出来了。在产品出入库管理这两个模块中问题表现最为突出，有时候一个查询要等较长时间才能完成。针对这一问题开发人员在以下几个方面进行了调整：

(1) 数据库存储文件的配置。创建了一个用户自定义文件组 myfileGroup，在该组中创建表和索引，并对日志文件使用 RAID10 磁盘阵列，对系统表和操作系统

文件使用 RAID1 阵列，对用户表和索引使用 RAID10 阵列。自定义文件组磁盘视图如图 1 所示。

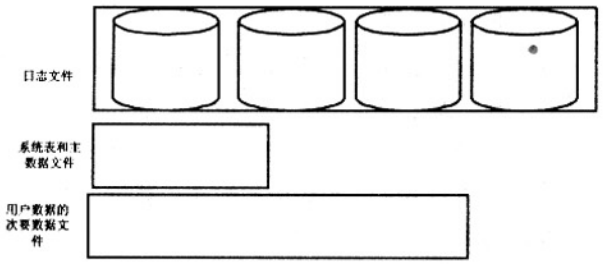

图 1　自定义文件组磁盘视图

（2）将原来 2 个大小为 18G 的磁盘分成 4 个 9G 的磁盘分区，将数据分布到这 4 个磁盘分区上。数据被分布到多个磁盘上后，被访问的数据页位于两个或多个磁盘的可能性极高，这样 I/O 可以在多个磁盘上同步（以并行方式）完成，从而大大地提高了性能。

（3）对系统的数据文件和日志文件的初始容量和再分配容量进行调整。初始容量太小时系统会要求再分配，以免产生碎片；再分配容量太小会导致频繁的页分配操作，产生很多的碎片，增加硬件搜寻数据时间。

（4）改进索引和查询语句。通过对系统索引的重新设置和查询语句的改进，系统的查询速度明显得到了提高，现将改进前后的情况通过表 1 和表 2 给出。

表 1　索引使用

聚簇索引和非簇集索引		
查看某段时间的出货信息 Select * from buy_infowhere buy_date > = '20070101'		
测试条件	测试结果	原因分析
buy_date 上有一个非聚集索引	211 毫秒	数据在物理上随机地存储在数据页上，查找时要执行一次表扫描
buy_date 上有一个聚集索引	147 毫秒	数据在物理上顺序地存放在数据页上，查找时先找到范围的末点，然后只在这个范围内扫描表
结论：在经常有范围查询（如 between，>，<，order by group by）发生的列上建立聚簇索引		
组合索引和前导列的选择		
查找某一型号的产品型号、名称和编号 select acc_num，acc_name，acc_id from acc_info where acc_num = ´GB893.1－86´and acc_name = '轴´		
acc_id，acc_name 上建立组合索引	203 毫秒	前导列是 acc_id 而这里没有引用它，因此没有利用上索引
acc_num，acc_name，acc_id 上建立组合索引	81 毫秒	前导列是 acc_num，使用了索引，形成了索引覆盖
经常同时存取多列，且每列都含有重复值，可以考虑建立复合索引来覆盖一个或一组查询，并且把查询引用最频繁的列作为前导列。		

表 2　查询语句构造

低效的 sql 语句	更改后的 sql 语句	更改原因
查看重庆和成都两地客户信息 select * from custom_info where cus_city = ´重庆´ or cus_city = ´成都´	select * from custom_info where cus_city = ´重庆´ union select * from custom_info where cus_city = ´成都	使用 or 可能造成索引失效，导致顺序扫描整个表，大大降低查询效率。
查找购买某型号产品超过 1000 的客户名称 select cus_name from custom_info where cus_id in（select buy_cusid from buy_info where acc_id = ' A00341' and buy_plannum > 1000）	Select cus_name from custom_info A inner join buy_info B on A. cus_id = B. buy_cusid where B. acc_id = ' A00341' and B. buy_plannum > 1000	对于主查询的每一条记录子查询都要执行一次，嵌套的层次越多效率越低
查找日平均销售量额超过 10000 的零件型号 Select acc_id from monthstock where surplus/30 > 10000	Select acc_id from monthstock where surplus > 10000 * 30	对列的任何操作结果都是在 SQL 运行时逐列计算得到的，因此就不得不进行全表扫描而不能使用列上的索引
查询某个月需要发出的产品类型和数量 Select distinct buy_type，buy_num from buy_info where buy_date between '20070101' and '20070131'	Select distinct buy_type，buy_num from buy_info where buy_date > = '20070101' and buy_date < = '20070131'	充分利用索引，非聚合表达式 between 将使索引失效

（下转第 14 页）

(上接第 10 页)

6 结束语

数据库性能优化是一个系统工程,应该从不同的方面同时考虑,不能孤立地考虑某一方面。本文从数据库设计优化、索引优化、sql 语句优化几个方面对企业级大型数据库性能优化提出了优化策略,旨在为系统设计和开发人员提供相对全面的参考。

参考文献

1 王俊伟、史创明,数据库管理与应用,北京:清华大学出版社,2006.7,50 - 54.

2 JEFFREY A. HOFFER, MARY B. PRESCOOT, FRED R. MCFADDEN 著 ,施伯乐、杨卫东、孙未未 译,现代数据库管理,北京:机械工业出版社,2004.9,100.

3 EDWARD WHALEN 著, 武欣、何畅、罗云峰 译, SQL Server 2000 性能调整技术指南,北京:机械工业出版,2005.7,47 - 49.

4 肖桂东,SQL Server 疑难解析,北京:电子工业出版社,2003.12,53 - 55.

5 DENNIS SHASHA, PHILIPPE BONNET 著, 孟小峰、李战怀 译,数据理与技术,北京:电子工业出版社,2004.5,67 - 70.