Sybase SQL Server 性能优化技术初探

袁长河 邢昭 吴永明 (上海同济大学计算机科学与工程系 200092)

摘要:分析了影响 Sybase SQL Server 性能的因素。在此基础上介绍了调优 Sybase SQL Server 的一些方法和技术。

关键词:性能优化 规范化 聚簇索引 填充因子

随着计算机应用的广度和深度日益深入,规模的不断扩大,以数据结构和算法设计为核心的、传统的结构化程序设计技术已显得有些不相适应。现代应用所面临的主要问题之一是:如何有效地管理和利用作为信息时代中不断产生和急剧膨胀的数据。现今的一些数据库管理系统(DBMS)虽提供了解决这一问题较好的手段,但要获得更好的、适应实际应用需要的效果,往往还需对其性能进行优化。本文就 Sybase SQL Server 性能的优化技术进行了一些尝试。

Sybase SQL Server 的性能优化犹如立体声收音机的调谐一样,调得过高或过底,均得不到期望的音响效果。要想得到比较理想的音响效果,应把注意力集中于可以产生最大的性能增益方面。对于 Sybase SQL Server 来说,影响它性能的,主要有以下一些因素和方面:

(1)数据库和应用的设计方面。这包括表的设计、表的数量和大小、表的设计规范化程度、事物的设计、identify 功能的使用、游标的使用等等。

(2)索引的使用方面。包括索引的数目及其在表上的分布和类型(聚簇或非聚簇、位逻辑等)、键的使用、填充因子的利用、唯一性、约束的使用、索引页面利用率、分布和密度统计的精确度以及对查询的适用性。

(3)查询构造方面。包括选择标准、操作符、函数、表达式和子查询、聚集函数、排序和分组等。

(4)过程对象的使用方面。如,规则、触发器和存储过程等。

(5)操作环境方面。包括内存的利用、磁盘的使用、CPU 的使用、网络的使用等。

一般来说,对上述这些不同的方面进行 Sybase SQL Server 的调优,会产生不同的效果。下面将介绍调优 Sybase SQL Server 性能的一些技术和方法。

一、表的优化设计

数据库的逻辑设计,包括表与表之间的关系,是优化关系数据库的核心。好的逻辑数据库设计可以为后面建立最优的数据库和应用性能打下基础。

1. 规范化逻辑数据库设计

规范化逻辑数据库设计涉及使用规范的方法把数据划分成多个相关的表,有比较多的狭窄表(即,具有较少列的表)往往是规范化的数据库的特征。合理地进行规范化常常能改善系统的性能。一般规范化有以下一些好处:

(1)由于表比较狭窄,排序和索引的建立就会更快些。

(2)由于表比较狭窄,表的数目就相对较多,有利于更多的聚簇索引。

(3)更窄更紧凑的索引。

(4)由于多个表较少的索引,能帮助提高 INSERT、UPDATE 和 DELETE 的性能。

(5)更少的 NULL,和更少的冗余数据,增加了数据库的紧凑性。

随着规范性的增加,获取数据所要求的连接数目和复杂性也随之增加。太多的表与表之间的复杂关系连接会妨碍系统的性能。合理的规范化应不使用多于四向连接的常用查询。在建表时,要有选择性地把一个大表规范成几个较小的表。

2. 定义键

定义主键、外键和"侯选"键是规范化的重要部分。以下是规范化数据库常用的一些做法:

(1)表的列依赖于表的主键。

(2)一个表的外键与另一个表的主键或"侯选键"关联。

(3)相关表中的行使用关系连接检索

对于经常连接的表，建议用于指定连接的列有相同的基数据类型，不包括可空值性或长度。

3. 有选择地采用非规范化方法

根据物理约束，有选择地采用一些非规范化方法可以提高性能。下面给出一些非规范化的措施及可能获得的增益。

(1)适当增加冗余列。可降低连接字数，减少锁定。

(2)适当增加派生列。如，计算列等。可减少表的连接数，并有效降低计算所占用的 CPU 时间。

(3)适当增加包含派生数据的表。有可能降低连接数目和代价，并减少连接处理所用的 CPU 时间。

但要注意的是，不恰当的非规范化做法可能会导致空间的浪费、增加数据同步修改的复杂性、查询 I/O 次数的增多等不利情况的产生。因此，要仔细权衡利弊。

二、索引优化技术

根据某个特定数据项的取值，索引的目的是让 SQL Server 的优化器能有一条到达该数据项的最有效的途径。在适当的地方采用恰当类型的索引对 SQL Server 性能的优化是很有帮助的。

1. 更新统计

每个索引均具有一个维护其统计信息的分布项。优化器利用这个项来判断该索引对某个特定的查询是否有用。但这个统计信息并不动态地重新计算。这意味着，当表的数据改变了之后，统计信息有可能是过时的，从而影响优化器追求最优工作的目标。因此，对于更新比较频繁的表，应定时运行其更新统计。

2. 采用唯一索引

第一键列值为条件的查询将通过最少次数的 I/O，就可找到期望的行。相反，若用于索引键的列有大量的重复值，则查询在找到期望的行之前，也许需要访问若干页面。因此，如果确信索引键的所有值都是唯一的话，则将索引定义成唯一的，以便优化器能利用这种高选择性。

3. 选择索引键

应选择那些采用小数据类型的列作为键，以便每个索引页能够容纳尽可能多的索引键和指针。通过这种方式，可使一个查询所必须遍历的索引页面降到最小。同时设法使索引的第一个列具有最大的选择性，因为当查询说明的搜索参数与索引的第一个列匹配时，该索引成为有用的。

4. 利用聚簇索引

聚簇索引的最低级——叶级，就是表页本身。不象非聚簇索引那样是指向表页的索引层。因为叶级的键值是数据行，所以数据行在物理上是顺序的。那些包含范围检查(BETWEEN，<，>等)或使用了 GROUP BY、ORDER BY 的查询可以很好的利用这些预排序的数据，达到减少或避免排序的代价。

5. 利用覆盖索引

覆盖索引是指那些包含了多个列的非聚簇索引，以便能覆盖一个或多个典型的查询。覆盖查询是指涉及的值存在于一个覆盖索引的键值中的查询。覆盖索引可以提高覆盖查询的性能，这种查询仅仅通过读取页面就可解决，节省大量的 I/O 开销。

三、优化查询

大多数的数据库活动都涉及到存取用户或系统对象的表页或索引项，SQL Server 存取页面的速度越快，其性能就越好。性能调优的很大一部分努力，就是要尽量减少页面存取，或使用内存的页面代替访问磁盘。下面是查询设计方面的一些技巧。

1. LIKE

当采用 LIKE 谓词时，如果以“%”或“－”等通配符开始时，则 SQL Server 将不能使用索引和分布页面。例如，应避免

```
>select * from employee where emp_name like "%son"
```

而宁可用

```
>select * from employee where emp_name like "s%"
```

2. 关系操作符和 BETWEEN

由于 B－树的遍历方式不适用于不等(！＝)比较，所以这些关系操作符不能用于索引选择的优化，而需要扫描索引的整个页级来找到适合的值。

对于其余关系操作符，避免额外的 I/O 方法是使用那些等于符，如＞＝，＜＝或＝。而其他的，如＞或＜，则要求读一些不必使用的页面。如，应避免：

```
>select * from employee where age>20
```

而宁可用：

```
>select * from employee where age>=21
```

在第一个例子中，你要读所有其值等于 20 的索引页面只是为了找到值为 21 的开始处，而在第二个例子中，你直接从正确的地方开始。

注意，以下两个查询是不一样的：

>select * from employee where age between 20 and 25

>select * from employee where age between 25 and 20

第二行的查询不返回任何结果行。

3. NOT EXISTS 和 NOT IN

与！=类似，我们也需要避免使用查询的否定形式(NOT EXISTS 和 NOT IN)以及 COUNT，应尽可能设法使用 EXISTS。

4. 数据类型的匹配

查询最微妙的问题之一是在 WHERE 子句中对具有不同类型的列的比较企图。在下面的查询中，gpa 定义成非唯一索引的浮点列，则该索引不会使用：

>select name,gpa from students where gpa=4

因为对数字的搜索被解释成一个整数，为使这个查询能够利用该索引，采用

>select name,gpa from students where gpa=4.0

下面列出这些相似的数据类型在 WHERE 子句中的比较运算中是不兼容的：

。float 和 integer

。char 和 varchar

。binary 和 varbinary

5. 连接

缺省情况下，多于四个表的连接(joints)按每次四个处理。对每个表集，找到并保存最佳的外表，其余的组合则用来估计下一个次外表。有时你可以通过增加冗余谓词给优化器一些它不会考虑的选择。如，这两个查询是相同的，但后一个查询提供了更大的灵活性：

>select * from A where a=b and b=c

>select * from A where a=b and b=c and a=c

6. 表达式

若 WHERE 子句中存在一个代数表达式，则优化器也不能使用分布统计信息。如，应避免：

>select emp_name from employee where age*2>40

而下面这个查询将产生更好的查询计划

>select emp_name from employee where age>40/2

7. 聚簇函数

当计算一个列上的聚簇时，应包括一个搜索子句，即使它指定的是所有行。如

>select avg(income) from students where gpa>0.1

使用 gpa 上的索引。而下面这个查询不使用索引：

>select avg(income) from students

四、优化操作环境

除了获取快速硬件之外，你还可以留心操作环境来影响 SQL Server 的性能。

1. SQL Server 的优先级

许多操作环境根据 CPU 的使用状况来动态地调整进程的优先级，小心别让操作系统或其他用户把 SQL Server 进程的优先级降低。如果 SQL Server 的优先级降低，其性能将会降低。如果你想将 SQL Server 进程的优先级改变为较高的值，应确保没有置的过高，以至影响操作系统的性能，SQL Server 的运行首先得依赖于操作系统的很好运行。

2. 减少网络流量

经常，全面提高网络的最佳方案是减少你传送的网络包数。有时某些查询没有很好地优化排列，从而返回多于用户需要的数据。通过认真设计你的应用，你可允许客户应用只把数量的行返回作为预览，在需要的情况下再返回全部的结果。这样，只有在用户的请求数多于预览数时才有必要等待。

3. 网络配置

网络协议的选择，必须通过的路由器、网关和网桥的数量，以及诸如网络缓冲区等配置参数，都会对 SQL Server 的客户/服务器的通信性能产生很大影响。应确保网络已为 SQL Server 的传输进行了最优化的设置。

结束语

可见，SQL Server 要获得理想的调整，需要你对应用和硬件有透彻的了解，你需要利用给出的信息、经验以及你对系统的了解，来作出明智的设计，然后需要测试你的结果并进行相应的调整和修正。

参考文献

[1] R. 兰金斯. SYBASE SQL Server 11 参考大全. 北京：宇航出版社，1998

[2] Hazlehurst P. Sybase System XI 实用大全. 北京：清华大学出版社，1997

[3] Karen Hogoboom 著，赵海燕，苏渭珍译. Sybase 系统管理员手册. 北京：机械工业出版社，1998

(来稿时间：1999 年 6 月)