SYBASE
技术服务园地
连载（65）
SYBASE TECHNOLOGY SERVICE FIELD

解决方案

开发人员升级至ASE 15.0的10大理由（六）

（接上期）

7 积极聚合、重定位连接与历史归档

ASE 15.0中新增的两项DSS技术是“积极聚合”和“重定位连接”。当这些技术用在历史归档上时，能在联合视图（手工表分区）上体现出一些优势。

7.1 积极聚合

积极聚合通过在查询树中往下推送可能的聚合函数来工作。与数据检索操作类似，假设启动了allrows_dss优化目标和advanced_aggregation查询标准，在一个OLTP数据库中仅包含了少量的数据子集有一个报表服务器中历史数据库的代理表，其它数据都驻留其上。通常都会使用完全联合(union all)建立一个视图来包含两个数据集，为既需要访问当前数据和历史数据的查询服务。在ASE 12.5中，如果用户从OLTP端运行该查询，ASE将所有历史数据拖拽至OLTP服务器上，然后执行聚合。拖拽巨大的数据将耗费相当长的时间，需要tempdb中密集的空间以便能让工作表存放数据，并且不能利用任何索引，因为工作表没有索引，除非ASE通过使用重新格式化/store_index查询优化技术来创建一个。这对使用了group by的矢量聚合尤其痛苦，因为用来保持重定位数据的工作表中缺少索引。ASE 15.0转而将聚合函数推送至联合的每端，后续将结果合并成最终的聚合值。

表11 向下推送的聚合的处理方式

聚合	向下推送的变量
sum(col)	sum(sum(col))
count(col)	sum(count(col))
avg(col)	sum(sum(col)) / sum(count(col))
min(col)	min(min(col))
max(col)	max(max(col))

表11中，加粗的聚合函数是最终的聚合，而其余的表达式则被向下推送至查询的每一端。假设与先前OLTP历史归档的情形相同，现在同样的查询将运行得快很多，因为聚合会在每端执行，仅是被聚合的行才被返回进行连接。虽然用来保持已聚合行的工作表仍旧没有索引(除非使用了store_index)，但需要用来连接少数几行的时间就会大大减少。

它同样在手工表分区的联合视图上运行完美。例如，考虑例程数据库pubs2，只是数据量更真实，132 357条sales和135万条salesdetail记录。虽然并非是真实的量，但却已足够展示ASE 15.0在DSS查询优化和积极聚合方面的威力。考虑以下查询：

```
select year(date), month(date), avg(qty), sum(qty)
from sales, salesdetail
    where sales. stor _id=salesdetail. stor_id
    and sales. ord _num=salesdetail. ord_num
    group by year(date), month(date) order by year
(date), month(date)
```

有两种可能的场景。（1）是salesdetail的所有数据按照典型的数据库定义被包含在一张单独的表中。（2）场景是联合视图场景，数据按照下表12分布在不同的表中。

表12 数据分布表

年份	表	行数
1997及更早	salesdetail_1997	114
1998	salesdetail_1998	160 212
1999	salesdetail_1999	272 766
2000	salesdetail_2000	261 612
2001	salesdetail_2001	275 301
2002	salesdetail_2002	262 626
2003	salesdetail_2003	117 624
	总计	1 350 255

视图定义类似于：

```
create view salesdetail as
select * from salesdetail _1997 union all
select * from salesdetail _1998 union all
select * from salesdetail _1999 union all
select * from salesdetail _2000 union all
select * from salesdetail _2001 union all
select * from salesdetail_2 002 union all
select * from sales
```

正如前面部分所描述的，对salesdetail的数据插入和更新都可通过一个创建在视图上的替代触发器完成。单一表和联合视图方法之间的优化区别如下。

（1）ASE 12.5/ASE 15.0 allrows oltp。sales表在与salesdetail表连接时用作外部表。对于每行记录，

嵌套循环连接浏览在salesdetail表上的每个主键索引。对于联合视图，ASE为联合的结果创建一张大工作表然后与sales连接。ASE 12.5则通过使用工作表作为连接sales的外部表来完成，而ASE 15.0在工作表上创建聚集索引然后进行sales→工作表的连接，原先通常都是与salesdetail表一道完成的。ASE 12.5使用排序工作表进行聚合，而ASE 15.0使用内存哈希表来进行聚合，并在内存中进行最终的order by排序。

(2) ASE 15.0 allrows mix。因为salesdetail表的主外键索引是基于{stor_id,ord_num}的，与其采用嵌套循环连接的方式对索引树进行密集的逻辑IO，ASE则采用sales表和salesdetail表之间的(非排序)合并连接。对于联合视图，它首先将所有的数据联合至一张工作表中，然后在sales和工作表之间进行合并连接，因为联合数据不需要按照索引排序的方式。通过哈希矢量聚合与order by内存排序，聚合按照与上述同样的方式完成。

(3) ASE 15.0 allrows dss。对于单一表来说，积极聚合技术支持ASE 15.0将聚合推送至salesdetail索引的每个组，通过在索引上使用GroupSorted操作符访问，并将结果作为与sales表(非排序)合并连接的外部表。最后的哈希矢量聚合确保了向下推送聚合的结束。对于分区/联合视图，通过相同的GroupSorted和用在sales表上的内存哈希连接将聚合向下推送至每个联合。最终的聚合通过哈希矢量聚合和内存order by排序实现。

注意，在allrows_oltp和allrows_mix中，联合视图将形成联合数据的工作表，继而要么在后续进行表扫描(由于缺少索引)作为外部表(allrows_oltp)，或被排序并用做合并连接(allows_mix)的内部表。这是语义分区的优势方面，将在后面讨论。

表13 优化响应时间的区别列表

ASE/优化目标	单表	联合视图
ASE 12.5.4	1 726	13 030
ASE 15.0.3 allrows_oltp	1 816	42 550
ASE 15.0.3 allrows_mix	1 636	4 536
ASE 15.0.3 allrows_dss	946	833

表13中，响应时间的区别很有趣(时间以毫秒计算)。Allrows_oltp中产生的巨大差异是因为在工作表上创建索引造成的。但是应该注意到ASE 12.5与ASE 15.0做比较，allrows_mix(缺省)对单表来说都差不多快，而对联合视图来说却快了3倍。这是项很大的差异，如果基于安全或其它考虑因素(例如在前面讨论的防止修改)需要使用联合视图技术，对ASE 12.5来说，从用户的角度几乎很难忍受，因为13s是非常明显的滞后。但是，对使用缺省优化目标，且对合并排序进行了一些调优的ASE 15.0来说，效果是仅需要3 s左右。虽然如此，使用allwors_dss和积极聚合功能，ASE 15.0中的结果比ASE 12.5中的快两倍，而联合视图的方法没有滞后，因为其性能也基本相当(在合理的误差范围内)。

该项技术支持开发人员可考虑将包含千万至十亿行级别的大而笨重的表使用联合视图分割成多个表，或仅是简单地将数据卸载到历史归档中，无须担心任何显著的性能差异。

7.2 重定位连接

重定位连接技术是专为被分割于OLTP和历史归档之间的数据而实施的。考虑上述查询，在sales和salesdetail表之间有连接。例如，因为经常使用，较早的sales数据未被转移至归档，而归档使用的是较慢的磁盘技术(分级存储)。ASE 12.5.x使用两种不同的方法处理远程数据：通过将远程数据重定位至本地服务器，在tempdb中创建本地工作表或游标本地结果来与远程数据连接。

7.2.1 ASE 12.5.x通过临时工作表的分布式连接

在ASE 12.5中，分布式连接常常导致远程表内容被大量通过CIS拖拽并且被插入工作表中，然后被用于与本地表的连接。处理步骤是：本地(OLTP)服务器发送查询片断并传递至远程(归档)服务器；远程服务器返回大量归档数据；本地服务器将归档数据保存至大工作表中并执行本地连接。

问题是在历史归档的情况下，拖拽的数据量巨大，将导致密集的网络处理时间和没有索引的大工作表。图4为上述流程。

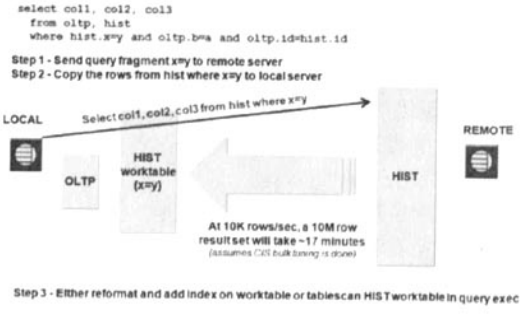

图4 ASE 12.5.x为远程数据使用本地工作表方法的分布式查询

除了时间因素以外，另一个主要的问题就是该方法需要tempdb有足够的空间保持来自远程服务

器的重定位数据。

7.2.2　ASE 12.5.x 通过游标的分布式连接

ASE 12.5.x用在分布式查询的另一个方法就是游标本地结果并使用连接键和查询片断来在远程服务器中寻找匹配的行。处理步骤是：本地(OLTP)处理所有能处理的本地查询(例如，如果包含了一张或多张本地表)，就如平常一样；对于本地查询所物化的每行，本地服务器将连接键和远程查询片断发送至远程服务器；远程服务器返回匹配的行，然后与物化行连接并将结果返回至客户端。本方法简单的图示如图5。

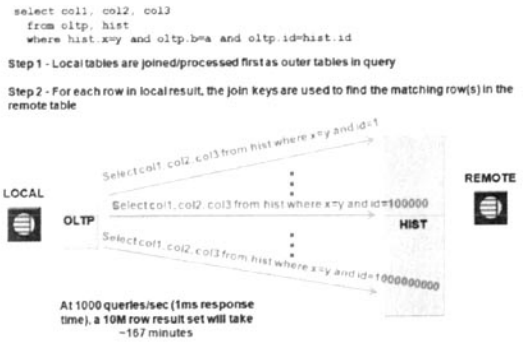

图5　ASE 12.5.x 通过游标方法的分布式查询

正如前面已经清楚证明了的，后面的这种方法对于大型结果集尤其慢。不幸的是，它看起来是最经常使用的方法，因为最常见的原因是缺少统计。由于缺乏对远程数据大小统计的了解，本地ASE只能猜测可能被影响的行数。在统计不是低得离谱的情况下，它常常假设不能做出正常的预估。在ASE 12.5.3之前，这几乎是难以逾越的问题，因为执行代理表的统计更新常常会因为tempdb空间的原因而造成失败或逐条运行，比预期的慢很多，因为它在远程服务器上手工执行select count(*)。该问题因为新增了将远程服务器系统目录的统计直接倒入本地服务器系统目录的功能而得到解决。

7.2.3 ASE 15.0.x 重定位连接

ASE 15.0使用重定位连接来工作。替代了OLTP服务器将所有历史数据进行拖拽，处理如下：本地(OLTP)服务器将查询片断发送至远程(归档)服务器；如果不存在的话，远程服务器动态创建较小代理表回本地服务器。如果代理表已存在，它使用导入的统计信息来决定如何将分布式查询片断处理回本地系统的好方法；在任何情况下，通过与远程服务器的代理表，本地服务器返回数量较少的数据（与12.5x比较）；远程历史服务器在本地处理连接，在大数据集上使用所有可用的索引和在较小表上已导入的统计，如果代理表已经存在的话；远程服务器返回结果至OLTP服务器并在本地服务器上通过传送模式传递给客户端，如图6。

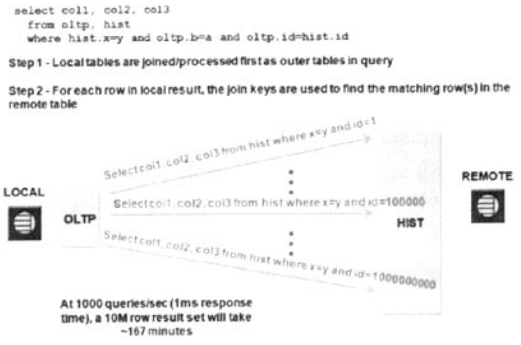

图6　ASE 15.0.2+ 分布式查询的重定位连接方法

虽然可能比所有的数据都在本地会慢，但从该技术中获取的性能足以支持重定位历史数据至ASE 12.5的归档数据库，即可避免性能损失。重定位连接有一些限制：本地和远程服务器必须是ASE 15.0.2或以上；应用程序不能使用客户端游标来检索结果。这也许是API的关系，因为一些API(例如，嵌入式SQL-C)需要游标；仅支持查询，不支持DML(例如包含连接的更新或删除)；引用text、image数据的查询将不会使用重定位连接；重定位连接必须通过sp_serveroption启动；两个服务器都需要有对方sysservers的入口。

该方法在远程服务器已经存在本地(OLTP)表的代理表，并且导入了这些表的统计情况下运行较好。

文/赛贝斯软件（中国）有限公司

（未完待续）

《铁路计算机应用》进入波兰《哥白尼索引》

经综合指标评估及国内外权威专家推荐，《铁路计算机应用》期刊已于2010年10月28日进入波兰《哥白尼索引》。（有关《哥白尼索引》介绍详见P63页）

文/本刊编辑部