表分区技术在短信增值业务中的研究与实现

Study and Application of Partitioned Tables in SMS Business

刘玉红　罗晓沛　（中国科学院研究生院 北京 100049）

摘　要：本文提出了利用表分区技术解决短信增值业务中海量数据的读写性能问题。详细论述了利用 SQL Server 2005 对短信增值业务中主要的大型数据表上行表和下行表的分区实现，并对表分区原理以及数据表的部署和分区表数据的拆分、合并与移动进行了介绍。实践证明，表分区技术的应用能显著提高大型数据表的读写性能和可维护性。

关键词：表分区　分区函数　分区架构　SQL Server 2005　短信增值业务

1　引言

随着移动通讯业务的迅猛发展，手机短信业务因其价格便宜、随时随地、方便快捷，获得了广大手机用户的青睐，成为了人们通信、交流与联络的常用手段。作为一个短信服务商，需要提供各种内容和类型的短信服务，面对每天上亿条的短信，如何保证高质量、及时地通信成为一个难题，本文主要从应用 SQL Server 2005 表分区技术入手来解决短信数据读写性能方面的难题。

2　表分区原理

超大型数据库的大小常常达到数百 GB，有时甚至要用 TB 来计算，而单表的数据量往往会达到上亿的记录，并且记录数会随着时间而增长，这不但影响着数据库的运行效率，也增大数据库的维护难度。表分区技术的合理运用可以使这些问题得到很大的改善，当表和索引变得非常大时，分区可以将数据分为更小、更容易管理的部分来提高系统的运行效率，如果系统有多个 CPU 或是多个磁盘子系统，可以通过并行操作获得更好的性能。

对表进行分区的主要目的就是改善大型表的可伸缩性和可管理性，是处理海量数据的一种十分高效的方法。根据特定的数据使用模式，使用定义的函数和架构对表进行分区，从而将具有相同分区键的所有行都直接放置到特定的位置，这样对表进行查询时，只需要在特定表分区内进行扫描，而不必进行全表扫描，缩短了磁盘检索时间，加快了查询速度。当把各分区分布在不同的物理磁盘上时，可提高这些磁盘的并行处理能力以优化查询性能。

3　SQL Server 2005 表分区实现方法

SQL Server 2005 中通过定义分区函数和分区架构对表进行分区设计，分区函数用于定义分区边界，创建分区函数后，则需要定义分区架构，便于将分区指向特定文件组，因为文件组允许将各个表放置到不同的物理磁盘上，这样可以最大程度的提高 IO 吞吐量，从而提高数据操作性能。创建分区表的逻辑流程如下：

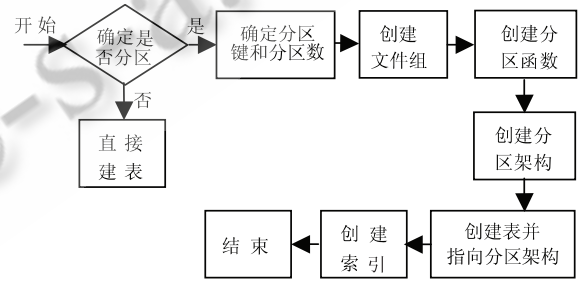

图 1　表分区设计逻辑流程图

本文以一个短信活动的上行和下行数据表的分区实现为例，因为是一个短期活动，但是每分钟需要处理几十万甚至上百万的数据，并且在活动进行中需要根据手机号码对上、下行数据进行频繁的统计，所以此处以手机号段作为分区的标准。

3.1 创建文件组

文件组可以达到分离数据以提高性能的目的，一

般情况下,文件组数最好与分区数相同,并且位于不同的磁盘上,每个文件组可以由一个或多个文件构成,而每个分区必须映射到一个文件组,一个文件组可以由多个分区使用。

以上行数据表的分区实现为例,在上行数据表的设计时,采用每个分区对应一个文件组的方式,因为共有 11 个分区,所以需要创建 11 个文件组,下面是创建文件组的代码:

```
ALTER DATABASE SMSDB ADD FILEGROUP [FG1]
```

在创建文件组后,使用 ALTER DATABASE 将文件添加到该文件组,代码如下:

```
ALTER DATABASE SMSDB
ADD FILE
( NAME = NFG1',
FILENAME = 'D:\DB\FG1.ndf',
SIZE = 10G,
FILEGROWTH = 10G
) TO FILEGROUP [FG1]
```

按照以上代码,分别创建其他的 10 个文件组和数据文件,并且将它们分布在不同的磁盘驱动器上。

3.2 创建分区函数

表进行分区的标准是通过分区函数来决定的,它定义 SQL Server 如何对数据进行分区,但这个操作并不涉及任何表格,只是单纯的定义了一个函数来分割数据,创建数据分区函数有 RANGE " LEFT | RIGHT" 两种选择。

此处分区将在上行表 MO 的 Mobile 列上进行定义,其数据类型为 Bigint ,因为分区键必须与分区列具有相同的数据类型,所以此处分区使用 Bigint 类型进行,并采用下边界 RIGHT 定义,下面是根据手机号段定义的分区函数代码:

```
CREATE PARTITION FUNCTION SMSDBPF(Bigint)
AS
RANGE RIGHT FOR VALUES
('13100000000', '13200000000', '13300000000',
13400000000', '13500000000', '13600000000',
13700000000', '13800000000', '13900000000',
14000000000')
```

在以上函数中,根据手机号段进行分区,因为还包含左侧和右侧两个边界,所以共创建了 11 个分区,分区范围是(-∞, 13100000000),[13100000000, 13200000000),[13200000000, 13300000000),……,[14000000000,∞)。

3.3 创建分区架构

分区函数必须和分区架构相关联,分区架构定义了分区函数所确定的分区放在哪个文件组中,虽然可以把所有的分区都放在默认的文件组 Primary 内,但是这不仅得不到性能的提升,而且不易于维护和管理,所以要把不同的分区放在不同的文件组中,创建分区架构的代码如下:

```
CREATE PARTITION SCHEME SMSDBPS
AS
PARTITION SMSDBPF TO
(FG1, FG2, FG3, FG4, FG5, FG6, FG7, FG8, FG9,
FG10,FG11)
```

在 SQL Server 2005 中,没有将分区函数和分区架构连接到任何数据表,这就增加了分区函数和分区架构的可复用性,也就是可以多个分区架构共用一个分区函数,多个数据表共用一个分区架构。

3.4 创建分区表

建立好分区函数和分区架构后,就可以创建分区表了。分区表是通过分区键值和分区架构相联系的,插入记录时,SQL SERVER 会根据分区键值的不同,通过分区函数的定义将数据放到相应的分区。下面代码是利用分区架构来创建短信上行表 MO,并在创建该表的 ON 子句中指定分区架构。

```
CREATE TABLE MO
(Mobile Bigint Not Null,
Content Varchar(200) Not Null,
……
ReceiveTime Datetime Not Null)
ON SMSDBPS(Mobile)
```

3.5 分区索引

除了对表进行分区外,还可以对索引进行分区。当索引和表使用相同的分区函数和列时,表和索引将对齐,如果表和索引不仅使用了相同的分区函数,并且还使用了相同的分区架构,则这些表和索引将被认为是按存储位置对齐,这可以使相同边界内的所有数据都位于相同的物理磁盘上,这样,在多 CPU 的情况下,每个处理器都可以直接处理特定的文件或文件组,并行运行多个进程,而不会与数据访问产生任何冲突。

对表增加索引虽然可以提高检索速度,但是同时也增加了维护索引的开销,使写入数据的效率也因此受到影响。在短信系统设计过程中,为了使读写不相

互影响,采用了数据复制技术,将数据从写服务器复制到另一台数据库服务器,并使其作为读服务器,使数据的读写分离开。可以在两台服务器使用相同的表分区结构,并且避免在写服务器的数据表内建立索引,而在读服务器内根据需要建立相应索引,使服务器处理能力都得以提高,关于读写服务器的分布图如下:

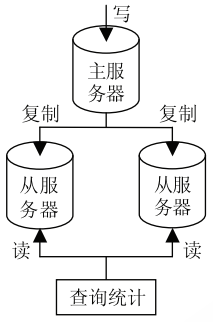

图 2　读写服务器分布图

创建分区索引的脚本如下:

```
CREATE CLUSTERED INDEX IX_MO
ON MO(Mobile)
ON SMSDBPS(Mobile)
```

3.6 查看分区表信息

在系统运行一段时间后,我们可以通过 $ Partition. SMSDBPF 来查看数据的具体分布情况,如查看每个分区存放的记录数,每个分区的内容等,具体代码如下:

```
SELECT  $ Partition. SMSDBPF(Mobile)
        AS[Partition Number]
      , Count(*) AS[Rows In Partition]
FROM dbo. MO
GROUP BY  $ Partition. SMSDBPF(Mobile)
ORDER BY [Partition Number]
```

以上查询执行结果如下图:

	Partition Number	Rows In Partition
1	1	4035045
2	2	3923575
3	3	4319935
4	4	4131305
5	5	3402780
6	6	6679330
7	7	6187415
8	8	5998540
9	9	6519555
10	10	6346485
11	11	5032935

图 3　分区表数据分布

上图显示了每个分区中数据的分布情况,下面我们对查询的执行情况进行研究,以查询某号段手机号码为例,通过 SQL Server 的显示估计的执行计划查看查询的执行情况,执行的代码和执行计划如下:

```
SELECT *
FROM MO
WHERE mobile >=13100000000
and mobile <13300000000
```

执行计划如下图:

图 4　执行计划

将光标悬停在图中所显示的" Constant Scan" 上,会发现它显示了参数值(((2)),((3))),这代表分区号,可以发现,针对以上查询,数据库只对这两个分区进行了查找,而不是整个大表。

3.7 性能对比

测试环境如下:

服务器型号:IBM3650

CPU:3.7G * 4

内存:16G

硬盘:136G * 2

操作系统:Windows 2003 SP2

数据库:SQL Server 2005 SP2

测试结果如表 1:

表 1　未分区和分区后性能对比表

条　件	未分区(ms)	分区(ms)
从 5000 万条记录中检索某号码的全部数据	24927	1163
使用 BULK INSERT 写入 30 万条数据	8386	4985

从表 1 可以看出,通过对表进行分区后,避免了对大

表的全表扫描,极大地提高了 CPU 和磁盘的 IO 并行操作能力,使数据的读写效率比未分区时有了明显提高。

3.8 数据移入移出

以上是针对一个短期短信活动设计的分区方案,对于常年运行的固定业务,则需要按照另外的规则进行分区,大多数情况下,只需要对最近两年的业务数据进行查询,这种情况下,可以按照时间对大型表进行分区,然而随着时间的增加,为了始终保持两年的数据,又不要因为添加或删除大量数据造成不必要的数据阻塞,就需要用到表分区的拆分、合并和移动。

当数据被移动到只读表或从只读表中删除,操作的代价变得十分高昂,不仅花费时间、占据日志空间,通常还会导致系统阻塞。表分区处理数据拆分、合并和移动的核心思想是,所有数据添加和删除操作都只是元数据的操作。其中拆分和合并是通过 ALTER PARTITION FUNCTION 进行,移动则是通过 ALTER TABLE 进行,在一切准备工作就绪后,数据处理工作可以在几秒内完成。具体操作的主要步骤是:

(1) 创建将要移入的分段表

在将要移出的分区的文件组创建分段表,然后将数据加载到分段表,为分段表建立索引和约束。

(2) 创建将要移出的分段表

创建第二个分段表,用于存储移出的分区中的数据,建立与要移入表相同的索引。

(3) 将旧数据移出分区表,并将新数据移入分区表

移出旧数据,更改分区函数以删除旧的分界点,将此文件组作为下一个使用的分区,更改分区函数添加新的边界点,更改基础表约束,移入新数据。

(4) 删除分段表

新数据已经被移入新的分区,分段表将不再需要,所以可以将分段表删除。

(5) 备份文件组

根据备份策略的不同,可以对数据库进行完全备份或者差异备份,但是对于大型数据库进行文件组备份更灵活,易于维护和恢复。

4 结论

本文对解决短信增值业务中的大型数据表的读写问题进行了详细的论述,并且通过实践证明,利用表分区技术很大程度的提高了读写的性能,下面对使用表分区的一些好处进行了总结:

(1) 利用表分区技术,将一个大表水平分区成若干小的分区,查询时只需访问某些分区数据,减少了对大表的全表扫描,缩短了磁盘检索时间,加快了查询速度。

(2) 将数据加载到现有分区表的新分区中时,最大减少了对其他分区中的数据访问的影响,并且性能相当于将同样数据加载到新的空表中。

(3) 允许通过将分区移入和移出分区表来维护分区,当删除和增加一个分区时,最大程度的减少了对其他分区的访问影响。

(4) 对数据表进行分区后,利于数据的备份与恢复,并且可以灵活的分配磁盘使用,极大地提高了磁盘 IO 性能,便于管理。

参考文献

1 Delaney. K Inside Microsoft SQL Server 2005:The Storage Engine. 1th ed. Bei Jing: Publishing House of Eletronics Industry, 2007. 288 – 295,162 – 173.

2 H. Inmon. W. Building the Data Warehouse. 4th ed. Bei Jing: China Machine Press, 2007. 35 – 40.

3 Knight,B Patel,K Snyder,W Armand,J – C LoForte,R McGehee,B Wort,S Salvatore,J Ji. H SQL Server 2005 高级管理. 1th ed. 北京: 人民邮电出版社, 2008. 333 – 337,423 – 434.

4 Nielsen. P SQL Server 2005 宝典. 1th ed. 北京: 人民邮电出版社, 2008. 650 – 688,933 – 942.

5 J. C. Mackin,Mike Hotek. SQL Server 2005 数据库服务器架构设计. 北京:清华大学出版社,2007. 269 – 304,404 – 414.