表分区在分界开关监控系统数据库的应用①

李亚龙, 朱 岩

(南京理工大学 自动化学院, 南京 210094)

摘 要: 针对分界开关监控系统数据库实时状态表数据量急剧增大造成查询效率降低并且维护困难的问题, 提出了对此类表进行分区的优化方案. 详细阐述了表分区优化处理的过程, 并通过分区窗口的滑动对实时状态表过时数据进行删除. 通过大量测试数据, 对比了分区前后的查询时间. 测试结果表明, 采用表分区技术能够显著提高查询效率, 改善数据库性能.

关键词: 数据库性能优化; 表分区; 索引分区; 监控系统

Application of Table Partition in Boundary Load Switch Monitoring System Database

LI Ya-Long, ZHU Yan

(School of Automation, Nanjing University of Science and Technology, Nanjing 210094, China)

Abstract: In face of problems such as low query speed and difficult maintenance caused by data volumes increasing in real-time state table of boundary load switch monitoring system database, this paper proposes data table partition optimization method, expounds the implementation process in detail, and removes obsolete data by sliding partitioned window. Through a large number of test data, the query efficiency of a same table before and after partitioning are compared. Results indicate that using the technology of table partition can shorten data query time dramatically and improve the stability of the database.

Key words: database optimization; table partition; index partition; monitoring system

在分界开关监控系统中, 远程终端(RTU)采集分界开关状态数据以一定频率上传到数据服务中心数据库, 用户通过客户端访问数据服务中心实现对分界开关状态的监控. 目前该系统已有近 2000 个 RTU, 上传状态数据的频率为 1 分钟, 每天将有 280 余万条记录, 每月将有 8400 多万条记录, 随着系统的运行, 加上不断有新的 RTU 添加到系统中, 实时状态表数据量将会迅速增加并变的异常庞大. 数据量的增加直接影响数据库响应查询的能力, 并且造成数据库备份、删除过时数据等维护操作困难, 进而影响整个系统的稳定性. 由此, 数据库的性能优化越来越凸显其重要性. 数据库的查询效率直接影响上层应用的响应时间和用户对上层应用软件的评价[1,2].

针对分界开关监控系统数据库面临的实际问题, 本文以 SQL Server 2012 为数据库平台, 深入研究了能够提高查询速度的表分区技术, 对其实现步骤作了详细介绍, 并利用分区窗口的滑动删除过时的历史数据. 最后, 通过分区前后的实验测试对比, 验证了分区效果.

1 表分区

表分区是一种物理数据库设计技术. 在一个大型数据库中, 数据库空间绝大多数是被少量表所占有[3], 可以考虑把一个数据量大的表按照查询特点拆分成多个小表来提高查询性能, 因为查询大多依据特定列的某一区间, 这样查询过程中只在特定的一个或几个分区浏览少量数据, 自然提高了查询速度. 此外, 如果具有多个 CPU 的系统处理一张大型表, 对表进行分区可以通过并行操作获得更好的性能.

表分区将表拆分为多个子表, 从物理上看, 这多

① 收稿时间:2015-05-17;收到修改稿时间:2015-06-08

个表可以存储在不同的位置，从逻辑上看他们仍为一个整体的表，如图 1 所示. 在 SQL Server 2012 中表分区分为水平分区和垂直分区. 水平分区每个表含有与原表相同的列数，垂直分区则是将原表分成多个包含较少列的表.

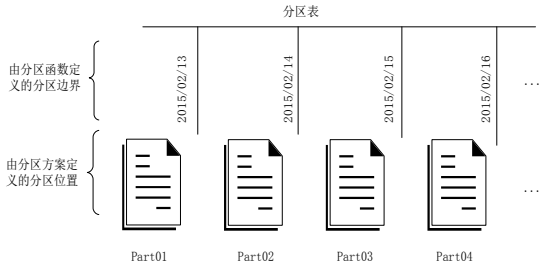

图 1　分区表示意图

使用分区提供了如下性能方面的优点[4]：①可以快速、高效地访问数据，因为只访问数据子集；②分区索引小了许多，因此在一个或多个分区上维护操作花的时间就少，于是可以降低重建索引的性能成本；③均衡 I/O，可以把不同的分区映射到磁盘以平衡 I/O，改善整个系统性能；④WHERE 子句的选择性常常会提高，对分区对象的查询可以只搜索特定的分区，提高检索速度.

2　分区实现步骤

分界开关的实时状态数据是以时间顺序进行存储的，实时状态数据表结构如表 1 所示. 用户对系统实时状态表使用频繁的功能主要有：对分界开关实时状态的查询、对分界开关近两个月内历史趋势的查询、对分界开关两个月内历史报警信息的查询. 可以发现这些查询都要提取近两个月内的某段时间区间的历史数据. 根据用户对实时状态表的使用情况，本文考虑将实时状态表按照记录的上传时间进行水平分区，每一个分区包含一天的数据，这样查询的数据集中在特定的分区，从而提高查询效率. 由于系统要求数据库只保留最近 60 天的数据，所以随着时间的推移，要求每天新增加一个分区，同时删除最左边的分区及其里面的数据，本文利用分区窗口的滑动删除过时的历史数据.

表 1　实时状态数据表结构

列名	数据类型	备注
Device_ID	CHAR(11)	设备编号
InfoTime	DATETIME	上传时间
I_A	FLOAT	A 相电流
I_C	FLOAT	C 相电流
I_Z	FLOAT	零序电流
V_A	FLOAT	A 相电压
Alert_ID	INT	报警编号

2.1 创建分区函数

分区函数指定表或索引如何划分. 创建分区函数时，应定义分区的个数，分区的依据列以及每个分区列的值范围，通过分区函数可以将表或索引的行映射到特定的分区. 根据分区函数的创建语法，LEFT|RIGHT 指定当间隔值由数据库引擎按升序从左到右排序时，分区边界值属于每个边界值间隔的那一侧(左侧还是右侧). 本系统中要求保留最近 60 天的数据，所以分区边界值从当前日期开始向前共有 60 个，将表分为 61 个分区，其中第 1 个分区没有数据. 创建分区函数的 SQL 语句如下:

```
CREATE PARTITION FUNCTION pfDaily (DATETIME) AS RANGE RIGHT FOR VALUES(N‘2015-4-17’,N’2015-4-16’,…,N’2015-2-17’)
```

2.2 创建分区方案

分区方案又称为分区架构，它构建在分区函数上，将每个分区映射到具体的文件组，指定了分区的物理位置. 实时状态表分区方案创建如下:

```
CREATE PARTITION SCHEMA psDaily AS PARTITION pfDaily ALL TO ([PRIMARY])
```

2.3 创建分区表

有了分区方案后，就可以创建使用它们的表和索引了，分区表是通过定义分区键值和分区方案联系的[5]. 插入记录时，SQL Server 会根据分区键值不同，通过分区函数的定义将数据放到相应的分区，从而将分区函数、分区方案和分区表、分区索引有机结合起来. 创建分区表的方法和创建普通表的方法相同，只是在指定文件组时指定分区方案和用于分区方案的列即可. 创建实时状态分区表和分区索引的 SQL 语句如下:

```
CREATE TABLE RealtimeState
(
    Device_ID CHAR(11) NOT NULL,
    InfoTime DATETIME NOT NULL,
    IA FLOAT NULL,
    …
)ON psDaily(InfoTime)
```

```
CREATE CLUSTERED INDEX IX_InfoTime_DeviceID
ON RealtimeState
(
    Infotime ASC,
    DeviceID ASC
)ON psDaily(InfoTime)
```

在删除 60 天前的数据时，需要借助一个辅助表来接收从实时状态表删除的数据，故还需创建一个辅助表 RealtimeState_Aux，该辅助表和实时状态表结构完全相同，无需进行分区.

2.4 分区窗口滑动管理

分界开关实时状态表每天都在增加数据，同时系统要对过早的历史数据予以删除. 所以要增加新的分区以接收新数据，删除过时的分区及其数据. SQL Server 2012 允许通过修改分区函数和分区方案来合并分区或拆分分区，从而实现分区窗口的滑动. 本文将增加分区和合并分区做成两个存储过程: 在分区表最右边添加一个分区(usp_AddPartitionRight)和删除分区表最左边分区(usp_DeletePartitionLeft).

在存储过程 usp_AddPartitionRight 中，可由系统提供的分区视图求出现有分区边界值的最大值(@max_boundury_value)，在此最大值上加一天作为新的分区边界值(@new_boundary_value)，将最右边分区拆分为两个分区. 主要实现如下:

```
SET @max_boundury_value = CAST((SELECT
TOP 1 value from sys.partition_range_values
WHERE function_id = (SLECT function_id from
sys.partition_functions WHERE name = ‘pfDaily’)
ORDER BY boundary_id DESC) AS DATETIME)
SET @new_boundary_value =
DATEADD(day,1,@max_boundury_value)
ALTER PARTITION SCHEMA psDaily NEXT
USED [PRIMARY]
ALTER PARTITION FUNCTION pfDaily() SPLIT
RANGE(@new_boundary_value)
```

在存储过程 usp_DeletePartitionLeft 中，需要借助辅助分区表 RealtimeState_Aux 来接收从实时状态表 60 天前(也就是第二个分区)的数据，然后求出分区边界值的最小值(@old_boundary_value)，将最左边的两个分区合并成一个分区，最后将辅助表 RealtimeState_Aux 里的数据删除以便继续接收数据. 主要实现如下:

```
SET @old_boundury_value = CAST((SELECT
TOP 1 value from sys.partition_range_values
WHERE function_id = (SLECT function_id from
sys.partition_functions WHERE name = ‘pfDaily’)
ORDER BY boundary_id) AS DATETIME)
ALTER PARTITION SCHEMA psDaily NEXT
USED [PRIMARY]
ALTER TABLE RealtimeState SWITCH partition 2
TO RealtimeState_Aux
ALTER PARTITION FUNCTION pfDaily()
MERGE RANGE (@old_boundary_value)
TRUNCATE TABLE RealtimeState_Aux
```

由于系统每天都在接受大量数据，分区窗口滑动每天都要执行，为了便于自动化管理，可以借助 SQL Server 代理来实现. SQL Server 代理(SQL Server Agent)是一个数据库实例中独立于数据库引擎的 Windows 服务[6]. 它能按照用户定义的计划定期执行代理中定义的作业，作业包含一个或多个作业步骤，每个步骤都有自己的任务.

利用 SQL Server Management Studio(SSMS)很容易在 SQL Server 代理中创建分区窗口滑动管理作业. 它包含两个步骤，分别是在分区表最右边添加一个新的分区即执行存储过程 usp_AddPartitionRight，然后再删除最左边分区及其数据即执行存储过程 usp_DeletePartitionLeft.

创建好作业步骤要创建计划. 计划是作业运行的时间，本系统每天都会增加和删除分区，所以该作业计划是每天的零点执行一次. 这样，新增数据会自动存储在分区表的新分区中.

3 分区前后的查询测试

为了检验表分区后带来的数据查询性能的变化，本文做了大量测试. 所选取的测试表，单表记录数达到 3300 多万，每个分区的数据量达到 50 多万. 表 2 展示分区后 61 个分区每个分区的数据量(未完全列出).

表 2 分区表每个分区的数据量

分区编号	记录数
1	0
2	5637724
3	5632705

4	5631915
…	…
60	5633007
61	5632885

本文在同一台机器上使用 SELECT 语句以特定 RTU 在某一时间区间为查询条件做了三组测试, 每组测试进行了三次, 测试结果如表 3 所示.

表 3 分区前后查询时间对比

时间区间	测试次数	未分区表查询时间/ms	分区表查询时间/ms
2015.02.18	1	470	76
	2	440	73
	3	433	66
2015.03.05-2015.03.10	1	543	173
	2	550	166
	3	546	156
2015.04.01-2015.04.15	1	596	306
	2	593	310
	3	590	306

由表 3 的查询时间可以看出, 分区后查询效率明显提升. 由表中三组的数据对比可以发现, 当查询时间区间比较大, 跨分区比较多时, 分区查询效率依然比未分区查询效率高, 但没有跨分区少时提升显著.

4 结语

本文针对分界开关监控系统数据库面临的问题, 为了满足用户对查询响应速度的需求, 保证系统的运行可靠性, 提出了表分区的优化方法, 探讨了表分区技术的具体实现步骤, 其中包括分区窗口的滑动管理. 分区前后查询速度的比较表明, 分区后对数据库历史数据查询效率明显提高. 该方法不仅可以用在分界开关监控系统数据库中, 也为其他监控系统数据库的设计与优化提供参考.

参考文献

1 陈俊.分区表及物化视图技术在查询分析中的应用[硕士学位论文].武汉:湖北大学,2010.
2 谷震离.关系数据库查询优化方法研究.微计算机信息,2006(15).
3 卢朝霞,习捷,王剑.数据库分区技术研究及应用.2006 全国光电子与光电信息技术学术研讨会论文集.2006:345–348.
4 徐述书,叶桦,仰燕兰.GPS 定位监控系统数据库的分区优化及其实现.东南大学学报(自然科学版),2011(z1).
5 詹利群,李涛.表分区技术在自动气象站数据存储中的应用.成都信息工程学院学报,2013(4).
6 秦婧等.SQL Server 2012 王者归来.北京:清华大学出版社,2014.