关系数据库中海量要素存储的分区优化研究

叶云霜,林伟华,刘福江,董晓莹
(中国地质大学(武汉) 信息工程学院,湖北 武汉 430000)

摘 要:目前,关系数据库中的分区技术应用相当广泛,但是用分区策略管理海量要素图层数据的存储与索引没有比较系统的技术方法。采用不同管理方式、不同分区粒度、不同索引方式及其组合的分区技术来系统地管理海量空间图层数据,进一步研究了不同的分区粒度及索引方式对查询效率的影响,并通过实验验证了关系数据库中的分区技术对海量要素图层数据的存储与管理具有优化作用。结果表明,在不使用分区键作为查询条件时,分区粒度越大查询效率越高;使用分区键作为查询条件时,本地分区索引查询效率更高等。利用合理的分区方案使得海量要素图层数据存储和管理得以优化,对矢量大数据的存储和管理研究具有重要意义,为更好地应用分区技术来解决实际遇到的存储与检索效率问题提供决策支持。

关键词:海量要素图层;关系数据库;分区;存储;索引
中图分类号:TP391.9 **文献标识码**:A **文章编号**:1673-629X(2020)01-0167-07
doi:10.3969/j.issn.1673-629X.2020.01.030

Research on Partition Optimization of Massive Element Storage in Relational Database

YE Yun-shuang, LIN Wei-hua, LIU Fu-jiang, DONG Xiao-ying
(School of Information Engineering, China University of Geosciences, Wuhan 430000, China)

Abstract: At present, the partitioning technology in relational database is widely used, but there is no systematic technical method for managing the storage and indexing of massive feature layer data by partitioning strategy. The partitioning techniques of different management methods, different partition granularities, different indexing methods and their combinations are used to systematically manage massive spatial layer data, and the effects of different partition granularity and indexing methods on query efficiency are further studied, and the relationship is verified by experiments. The partitioning technology in the database optimizes the storage and management of massive feature layer data. The results show that the query efficiency is higher when the partition key is not used as the query condition. When the partition key is used as the query condition, the local partition index query is more efficient. Using reasonable partitioning schemes to optimize the storage and management of massive feature layer data is of great significance for the storage and management of vector big data, and provides a decision for better application of partitioning technology to solve the storage and retrieval efficiency problems actually encountered.

Key words: massive feature layer; relational database; partition; storage; index

0 引 言

对于海量空间数据的高效存储、管理等问题的研究,已经发展了近半个世纪,国内外学者已经做了大量的研究工作,并取得了显著成果。如面向对象的关系型数据库,主要基于关系型数据库开发支持空间数据管理的扩展模块。再有就是空间数据引擎中间件技术,使得空间数据可以用关系型数据库进行存储并用于 GIS 平台上的各种复杂查询与分析,能够与特定 GIS 平台紧密结合,空间处理效率更高。后来,Thomas Kyte[1](2005)详细地阐述了关系数据库中分区的概念、策略、扩展及索引分区;目前关系数据库中的分区技术在管理关系型数据中应用较广泛,如徐畅等(2017)采用影像获取难易程度分区及统筹学的思想,设计第一次全国地理国情普查标准时点核准影像获取

收稿日期:2019-01-23 **修回日期**:2019-05-23 **网络出版时间**:2019-09-24
基金项目:国家“十二五”“863”项目(2014AA121401);湖北省 2014 年面上自然科学基金项目(2014CFB911)
作者简介:叶云霜(1997-),女,研究生在读,研究方向为摄影测量与遥感和海量空间数据存储;林伟华,副教授,硕导,通讯作者,研究方向为地图制图学与地理信息工程。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20190924.1535.038.html

及覆盖方案,实现对普查标准时点核准遥感影像任务区的优化覆盖[2]。而其在空间数据管理中应用较少,胡小彭等(2017)提出了一种基于分区存储技术的空间数据入库解决方案,并应用到安徽省地理国情普查中,验证了分区存储数据库的读取效率明显高于非分区存储数据库[3]。张小文等(2011)以 SuperMap 地理信息系统控件为基础,设计与实现了分区技术的灵活管理,与预报员交互的预报流程和预报结果的 Web 发布等功能[4]。但是,当矢量图层要素量达到 10 万条时,数据库查询性急剧下降,管理负担增大。如何利用关系型数据库的分区技术管理大容量的空间数据尤其是要素量达到千万级别以上的矢量图层数据,仍是困扰广大空间数据用户的主要问题之一。

因此,文中采用不同管理方式、不同分区粒度、不同索引方式的分区技术来系统管理空间大图层数据,并通过实验验证分区技术对海量要素图层数据的存储具有优化作用。

1 海量要素图层分区技术

数据分区技术是按照约定的方式从物理上划分库表结构,但分区表中的每个分区在逻辑上是独立的。如果选择合适的数据分区策略,会大大加快数据的查询速度[5-7]。

1.1 分区存储管理构建

由于分区表不能通过 ArcGIS 等 GIS 平台创建,因此需要先在数据库中建好表结构。构建过程如下:

(1)在 GIS 平台创建企业级地理空间数据库(create enterprise geodatabase),创建 SDE 用户、SDE 表空间、安装 SDE Repository(SDE 资料档案库,包含空间数据字典和 ArcSDE 软件程序包);

(2)在关系型数据库中创建数据存储所用表空间、用户并授权;

(3)在关系型数据库中创建分区表,并创建全局空间索引;

(4)在关系型数据库中创建分区表,并创建本地空间索引;

(5)按设置好的规则执行数据存储。

1.2 分区存储查询算法

将数据按照分区进行组织后,数据在逻辑上是同一张表,查询时表名没有区别,但实际数据存放在不同的分区中,可能存放于不同的服务器等存储设备上。在地图窗口进行查询浏览时是整张表同时展示在地图窗口中,那么在执行空间查询如框选时数据库可能同时需要访问多个分区,需要考虑是否先找到与感兴趣区域相交的分区再进行查询;再到分区中执行相关查询返回需要的结果,或者直接就只使用感兴趣区域作为查询条件。分区的算法有 3 种,分别命名为 part_query1、part_query2、part_query3,其伪代码分别为:

(1)part_query1。

根据 slfw(矢量范围底图),判断 BR(boundary rectangle,范围矩形)与哪些矢量底图相交,得到相交矢量范围列表 slfw_lists。将 slfw_lists 作为查询条件之一,SQL 语句样式如下:

“select shape from part_table t where slfwdm in (slfw_lists) sdo_filter (t. shape,BR))”;

(2)part_query2。

将得到的矢量范围列表 slfw_lists 的过程内嵌到 SQL 语句里,其样式如下:

“select shape from part_table t where slfwdm in (select slfwdm from slfw t where sdo_filter(t. shape,BR));sdo_filter(t. shape,BR))”;

(3)part_query3。

仅使用 BR 作为查询条件,SQL 语句样式如下:

Select shape frompart_table t where sdo_filter(t. shape,BR));

其中,part_query1、part_query2 都是先用矢量范围底图与范围矩形进行相交查询获取目标要素所在的矢量范围代码,然后再到图层元数据表中根据矢量范围代码获取目标要素所在子表名,然后再到相应的子表中做精准查找。而 part_query1 和 part_query2 的区别在于,part_query2 同样将表示在各个子表中查找的 sql 语句保存到一个总的 sql 语句中只执行一次在分表中的精准查找,尝试以此来提高效率;part_query3 则是利用分区表向上只有一个图名、逻辑上不分离的特性不考虑子表直接做大表与 BR 的相交查询得到目标结果要素。

1.3 分区策略技术方法

1.3.1 技术流程

最优分区方法的确定是通过采用不同分区机制来确立的。首先对存储海量要素图层数据的数据库大表进行分区拆分。其次,深入研究应用分区技术管理海量要素图层数据时的具体问题:分区键的确定、不同分区粒度的选择、空间索引的建立。然后,利用事件追踪查询时数据库内部的操作,结合前面的实验结果进一步研究不同的分区粒度及索引方式对查询效率的影响。最后,确定大表拆分方案。技术流程如图 1 所示。

1.3.2 分区键的选择

分区时需要选择一个属性项作为分区作用对象来完成表的分区,即分区键。分区键的选择需要考虑被访问次数最多的那些属性字段(例如属性字段“用途”访问次数最多,则把用途相似的要素放在同一分区中)、归档频率(归档数据显然使用时间字段实现范围

分区是较好的选择[8-10])等。

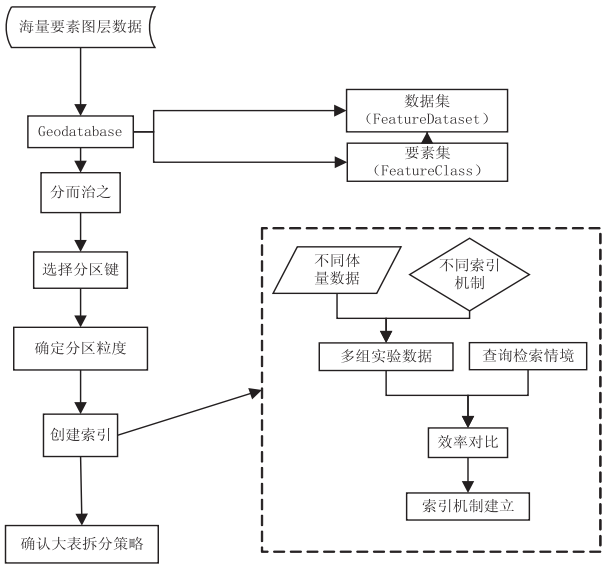

图 1　技术流程

1.3.3　分区粒度的选择

在确定了分区键之后,就需要考虑分区粒度的选择[11]。分区粒度的选择没有固定的模式或者原则,比如:便于系统维护考虑的是多大的分区能够维护方便,比如银行系统的用户信息存储,按市分区比按省分区后期维护方便,则采用用户信息存储按市分区;便于归档考虑的是归档频率,比如对小区用户管理,按单元楼归档比按门牌号进行归档方便,不同目的采用的分区方法不同。

1.3.4　索引机制的构建

在分区之后再对数据创建索引双管齐下能够快速提提升查询检索效率[12-14]。关系型数据库中管理空间数据使用较多的是四叉树索引和 R 树索引。与四叉树索引相比,R 树索引更自动智能,具有较高的存储利用率。

2　实验验证

2.1　实验环境

文中测试环境的硬件设备包括 GIS 应用服务器、数据库服务器、测试机。软件设备包括 Windows Server 2008、Arcgis[15]服务器 10.2.2 版本、Linux[16]操作系统、Oracle[17-18]关系数据库 11g、旗舰版 Microsoft Windows7 、IE 测试浏览器。

2.2　实验数据

为了验证数据大表拆分方案分区在包含海量要素的大图层(一般值要素量超过 200 万条图层)上的适用性,以贵州省 87 个县的地类图斑数据为研究对象,该地类图斑图层数据包含的要素总数为 6 695 554 条。查询时随机从贵州省省域内选择 3 个样本范围,作为空间查询时的查询范围。将 6 * 3 个样本范围分别与 3 个实验主体进行空间查询运算,记录每次查询的耗时。

2.3　实验分析

文中研究的海量要素图层,在其应用场景中,当在地图窗口中对这些矢量数据进行浏览或查询时,多数情况下其本质为对集中连片要素集合的访问,这些集中连片要素地理位置很接近,通常情况下归属于一个或几个行政单元。因此,考虑到若能够在进行矢量数据查询时,根据行政区化过滤和缩小数据库中检索范围,则能够起到访问较少数据而较快找到满足条件数据的效果;另外,在多用户并发访问场景下,同一时刻不同用户访问的数据通常是不同地理范围的,那么若不同地理范围的数据存放在不同的存储设备上,则能够起到 IO 负载均衡的效果。因此,文中选择以行政区代码作为分区键、将相同行政区代码的要素存放在同一个分区中的分区策略。

以地类图斑为测试数据,设计实验测试如何选择分区粒度。国内行政区划大致可分为:国、省、市、县、乡镇五个级别,而分区是将分区后的数据分配单独的段,国内有 40 446 个乡镇级别的区域,若将数据分成 40 446 个,会导致分区数据过于分散、细致,数据的管理和查询功能性能降低,验证该种分区方案必要性不高。因此,文中主要考虑验证“按省分区”、“按市分区”、“按县分区”这三种分区方式的效率。

为了模拟真实应用场景,提高查询检索效率,在对数据进行分区之后也为其创建空间索引,空间索引的类型均选择适合空间对象的 R 树索引。与表分区一样,索引也可以分区,索引分区方法有本地分区索引和全局分区索引。本地分区索引指的是每个表分区都有一个索引分区,而且只索引该表分区,是一对一的关系。全局分区索引指的是索引存储时整体按照范围或者散列进行分区。此时,一个索引分区可能指向任何(和所有)表分区,一个表分区的数据也可能存在于任何一个索引分区。此外,全局索引还可以不进行分区,整体存储在一个分区中。

2.3.1　不同分区方案查询效率分析

首先,为得到不同分区方案下的查询效率,将全国范围内的分别为千万级和亿级要素量的两组数据分别按照“未分区+空间索引”、“按县分区+本地空间索引”、“按县分区+全局空间索引”、“按市分区+本地空间索引”、“按市分区+全局空间索引”、“按省分区+本地空间索引”、“按省分区+全局空间索引”分为七组。对于分组后的数据分别选取 1 ∶ 500、1 ∶ 2 000、1 ∶ 10 000、1 ∶ 25 000、1 ∶ 50 000、1 ∶ 100 000 六个覆盖常用的大中小查询场景,针对全局和本地空间索引根据上述结论分别采用适合其的 part_query3(全局空间索

引)和 part_query2(本地空间索引)算法在全国范围内随机生成各比例尺的查询范围,在 Oracle 数据库中进行空间查询,记录查询耗时。最后,采用百分位数回归法[19]计算平均值和第 100 位百分位数,结果如表 1 ~ 表 4 所示。并对查询结果做归一化处理(用不同分区方案的查询统计结果与当前比例尺下查询耗时最低的值的比值表示)。

表 1 千万级要素量下不同分区方案查询效率平均值 s

比例尺	按县+本地	按县+全局	按市+本地	按市+全局	按省+本地	按省+全局	未分区
1∶500	0.281 576	**0.112 829**	0.223 738 6	0.144 981	0.162 84	0.141 631	0.137 758 1
1∶2 000	0.094 924	**0.032 333**	0.042 648	0.041 243	0.039 038	0.036 6	0.034 005
1∶10 000	0.854 919	**0.572 481**	0.643 586	0.592 871	0.683 252	0.596 217 1	0.644 51
1∶25 000	0.834 943	0.732 552	0.839 405	0.703 538	0.831 095	**0.668 272 4**	0.794 14
1∶50 000	1.544 852	1.232 281	1.374 138	1.324 457	1.450 357	**1.220 019**	1.405 067
1∶100 000	**11.834 27**	12.442 61	15.391 04	15.753 04	14.800 15	14.079 977	12.792 29

注:表中字体加粗部分圈出的为该行的最小值,表示该比例尺下查询耗时最短。

对查询效率进行排序:按县+全局>按省+全局>未分区>按市+全局>按省+本地>按市+本地>按县+本地。

表 2 千万级要素量下不同分区方案查询效率第 100 百分位数 s

比例尺	按县+本地	按县+全局	按市+本地	按市+全局	按省+本地	按省+全局	未分区
1∶500	0.537 367	**0.227 723 3**	0.461 733	0.323 2	0.373 767	0.331 967	0.303 733
1∶2 000	0.083 567	**0.030 6**	0.043 533	0.032 467	0.032 72	0.031 433	0.034 5
1∶10 000	2.763 8	**1.811 2**	2.165 7	1.946 333	2.345 867	2.016 767	2.336 8
1∶25 000	2.431 75	**1.970 467**	2.385 767	2.169 467	2.470 7	2.022 993 3	2.413 3
1∶50 000	3.952 767	**3.042 667**	3.628 33	3.239 533	3.398 57	3.243 11	3.621 33
1∶100 000	41.843 6	**39.857 83**	46.450 4	51.106 7	47.114 13	45.333 6	44.393 8

注:表中字体加粗部分圈出的为该行的最小值,表示该比例尺下查询耗时最短。

对查询效率进行排序:按县+全局>按省+全局>按市+全局>未分区>按省+本地>按市+本地>按县+本地。

表 3 亿级要素量下不同分区方案查询效率平均值 s

比例尺	按县+本地	按县+全局	按市+本地	按市+全局	按省+本地	按省+全局	未分区
1∶500	0.391 576	**0.208 829**	0.357 386	0.279 81	0.348 4	0.296 31	0.277 581
1∶2 000	0.139 924	**0.067 333**	0.089 648	0.075 243	0.085 038	0.079 6	0.076 005
1∶10 000	1.118 919	**0.842 481**	1.034 586	0.890 871	1.004 252	0.912 171	0.994 51
1∶25 000	1.084 943	0.872 552	1.059 405	0.918 538	1.071 095	**0.852 724**	1.011 514
1∶50 000	4.344 852	3.342 281	4.586 138	3.546 457	4.050 357	**3.230 019**	3.875 067
1∶100 000	**17.735 27**	20.142 61	23.301 04	23.893 04	19.766 15	22.699 77	20.572 29

注:表中字体加粗部分圈出的为该行的最小值,表示该比例尺下查询耗时最短。

对查询效率进行排序:按县+全局>按省+全局>按市+全局>未分区>按省+本地>按市+本地>按县+本地。

表 4 亿级要素量下不同分区方案查询效率第 100 百分位数 s

比例尺	按县+本地	按县+全局	按市+本地	按市+全局	按省+本地	按省+全局	未分区
1∶500	0.797 367	**0.477 233**	0.701 733	0.623 2	0.768 767	0.701 967	0.709 733
1∶2 000	0.293 567	**0.150 6**	0.218 533	0.202 467	0.217 2	0.202 433	0.194 5
1∶10 000	2.733 8	**1.851 2**	2.455 7	2.026 333	2.415 867	2.136 767	2.396 8
1∶25 000	3.137 5	**2.199 933**	3.098 767	2.739 467	3.130 7	2.340 467	3.053 3
1∶50 000	9.352 767	**5.631 1**	10.298 33	7.289 533	6.998 567	6.432 667	6.429 13
1∶100 000	63.183 6	**54.695 8**	93.420 4	76.062 67	68.715 13	73.013 6	57.087 8

注:表中字体加粗部分圈出的为该行的最小值,表示该比例尺下查询耗时最短。

对查询效率进行排序:按县+全局>按省+全局>未分区>按市+全局>按省+本地>按市+本地>按县+本地。

观察比较表 1 ~ 表 4 可知:

(1)对于文中的实验数据,“按县+全局”索引效率更高;

(2)全局索引比本地索引效率更高;

(3)索引为全局索引时,三种分区粒度的效率从高到低分别是:按县分区、按省分区、按市分区;

(4)索引为本地索引时,三种分区粒度的效率从高到低分别是:按省分区、按市分区、按县分区,分区粒度越大效率越高。

2.3.2　分区粒度对查询效率的影响分析

已知查询检索的要素集数据量越大则耗时越久,但在数据库中几百与几千甚至几万条记录查看时实际上相差甚小,因此只要合理选择分区粒度,这都不会是影响查询效率的主要原因,文中主要研究除此之外由于分区而额外增加的查询操作。从 elapsed 中的三种数据库调用类型 parse、execute、fetch 入手,利用 tkprof 分析 trc 文件。以按县分区查询中耗时较多的一系列 SQL 为参照,比较相同 SQL 在按省分区空间查询中的耗时。以数据字典表 Seg $ 查询为例,如表 5 所示。

表 5　Seg $ 表中查询过程统计分析

分区方式	call	count	cpu	elapsed	disk	query	current	rows
	Prase	0	0	0	0	0	0	0
按县	Execute	17 796	1.55	1.54	0	0	0	0
分区	Fetch	17 796	0.20	0.20	632	53 388	0	**17 796**
	tatal	35 592	1.75	1.75	632	53 388	0	17 796
	Prase	0	0	0	0	0	0	0
按省	Execute	383	0.04	0.04	0	0	0	0
分区	Fetch	383	0	0	48	1 150	0	**383**
	tatal	766	0.04	0.05	48	1 150	0	383

注:表中字体加粗的为执行次数。

SQL 查询的对象是 seg $,按县分区对 SQL ID 为 9tgj4g8y4rwy8 的 SQL 执行了 17 796 次,总耗时间为 1.75 s;而按省分区对相同 SQL 的执行次数仅为 383 次,总耗时为 0.05 s。在此 Lobfrag $ 、Obj $ 、Indpart $ 、Obj $ (2)、Tabpart $ 、SDO_FILTER 数据字典表不做赘述,得出:

(1)分区越多、表上的 lob column 越多,对数据字典表的访问次数越多。

(2)全局空间索引下,数据经过分区后,不同分区粒度之所效率会有不同,差异在于对于数据字典表的访问次数上,查询效率与分区粒度的大小呈负相关,在主要几个与分区相关的数据字典表中具有如表 6 的关系。

表 6　全局空间索引下各数据字典的访问次数与分区个数的关系

数据字典表	访问次数
Seg $	$(1+2Y)*X$
Lobfrag $	XY
Obj $	XY
Indpart $	XY
Obj $ (2)	XY
Tabpart $	X

其中,X 表示分区数,Y 表示 lob column 个数。

2.3.3　索引方式对查询效率的影响分析

索引也可以分区,为了研究索引分区对空间数据查询效率的影响,同样使用范围为全国的要素量在千万级别的数据进行实验,将数据分别按照按县和按省进行分区并且创建本地空间索引。同样开启 10046 事件[20](Oracle 中获取最完整的 sql 执行计划的一个功能),跟踪 SDO_FILTER 操作。使用 tkpof 分析 trc 文件中耗时最多的 SQL,对根据绑定变量的值分析不同 SQL(主要是对数据字典的递归查询)查询的数据内容。比较在使用分区索引时所查询的数据字典内容,

与在使用全局索引时查询的数据字典内容。在使用分区空间索引时,按县分区与按省分区对数据字典的访问次数及返回记录数如下。同样为了分析 SQL 语句解析次数与分区数的关系,将查询语句中的绑定变量提取出来与相应的数据字典表进行连接,以 Seg $ 表和 Obj $ 表的查询统计过程为例,见表 7、表 8。

表 7 Seg $ 表中查询过程统计分析

分区方式	call	count	cpu	elapsed	disk	query	current	rows
	Prase	0	0	0	0	0	0	0
按县	Execue	17 814	1.59	1.61	0	0	0	0
分区	Fetch	17 814	0.22	0.21	659	53 442	0	**178 140**
	tatal	35 628	1.82	1.83	659	53 442	0	178 140
	Prase	0	0	0	0	0	0	0
按省	Execue	395	0.08	0.07	0	0	0	0
分区	Fetch	395	0.01	0.01	50	1 185	0	**395**
	tatal	790	0.09	0.08	50	1 185	0	395

注:表中字体加粗的为执行次数。

查询内容包括 LOB INDEX PRTITION、LOB PARTITION、TABLE PARTITION、少量 sys 和 mdsys 用户下的表和其他表。

以按县分区表为例,包括:

INDEX PARTITION:2 531 * 3 = 7 593

此处不包括分区空间索引,仅包括 LOB INDEX PARTITION;

LOB PARTITION:2 531 * 3 = 7 593

TABLE PARTITION 为 2 531。

表 8 Obj $ 表中查询过程统计分析

分区方式	call	count	cpu	elapsed	disk	query	current	rows
	Prase	1	0	0	0	0	0	0
按县	Execue	10 125	1.17	1.15	0	0	0	0
分区	Fetch	10 125	0.10	0.12	198	40 500	0	**10 125**
	tatal	20 251	1.28	1.28	198	40 500	0	10 125
	Prase	1	0	0	0	0	0	0
按省	Execue	173	0.04	0.04	0	0	0	0
分区	Fetch	173	0	0	4	692	0	**173**
	tatal	347	0.04	0.04	4	692	0	173

注:表中字体加粗的为执行次数。

查询内容包括 LOB PARTITION、SPATIAL INDEX PARTITION 相关的 TABLE PARTITION。

以按省分区表为例,包括:

LOB PARTITION:43 * 3 = 129

INDEXPARTITION 为 43,此处包括 SPATIAL INDEX PARTITION,不包括 LOB INDEX PARTITION

TABLE PARTITION:(仅包括与查询范围相关的分区,可忽略不计)

Lobfrag $ 、Indpart $ 、Obj $ (2)、Tabpart $ 不做赘述,得出访问次数与分区个数的关系见表 9。

表 9 本地空间索引下各数据字典的访问次数与分区个数的关系

数据字典表	访问次数
Seg $	$(1+2Y)*X$
Lobfrag $	XY
Obj $	$(1+Y)*X$
Indpart $	$(1+Y)*X$
Obj $ (2)	$(1+Y)*X$
Tabpart $	X

其中,X 表示索引分区数,Y 表示 lob column 个数(与数据本身格式有关)。已知为所有的数据分别按

县和按省进行了分区并为其建立了全局空间索引。根据表 9 可知:

(1)对数据字典访问次数与索引分区的粒度线性相关,分区数越多,访问次数越多;

(2)索引分区与数据分区相似,均是在查询对象的数据量达到一定程度后,即索引分区是在要查询的索引池中数量达到影响查询效率时可以考虑进行索引分区;

(3)索引是提升查询效率最有效的方法。文中实验是在普通查询环境下(即不以索引主键为查询条件)的查询效率分析,如果查询条件中包含分区键使用本地空间索引将可以直接定位到相应的分区中,查询对象将仅限于那个子分区,在这样的查询情景下本地空间索引将会有更高的查询效率。

3　结束语

在前人利用关系型数据库的分区技术不能有效管理大容量的空间数据尤其是要素量达到千万级别以上的矢量图层数据的基础上,文中采用不同管理方式、不同分区粒度、不同索引方式及其组合的分区技术来系统管理空间大图层数据,并通过实验验证了分区技术对海量要素图层数据的存储具有优化作用。得出结论:(1)分区技术在对海量要素图层数据存储和查询中具有优化作用;(2)仅仅考虑查询效率时,如果在分区后各个子表中查询操作本身差别不大,在不使用分区键作为查询条件的查询情境中,分区粒度越大则效率越高;(3)仅仅考虑查询效率时,使用分区键作为查询条件时,则本地分区索引效率更高;否则全局空间索引在查询情景中表现更优等。该方法为更好地应用分区技术解决实际遇到的存储与检索效率问题提供技术支持。但是,用分区技术来管理其他类型的数据时,分区键的确定、分区粒度的选择和索引方式的确定不能一概而论,还应根据实际情况确定,这也是下一步研究的内容。

参考文献:

[1] KYTE T. Expert oracle database architecture:oracle database 9i,10g,and 11g programming techniques and solutions[M]. [s. l.]:[s. n.],2005.

[2] 徐　畅,雷　兵,张　涛. 地理国情普查时点核准影像保障设计与实现[J]. 测绘与空间地理信息,2017,40(9):158-161.

[3] 胡小彭,任家锋,魏雪梅. 分区存储技术在地理国情普查数据库中的应用[J]. 地理空间信息,2017,15(5):39-41.

[4] 张小文,张世强,陈良华,等. 长江三峡库区以上地区数值化降水预报系统的设计及实现[J]. 遥感技术与应用,2011,26(5):632-639.

[5] 雷　刚,周可法,张楠楠,等. 数据分区在地学空间数据查询中的应用[J]. 计算机应用,2010,30(S):148-151.

[6] CHEN Zhikun,YANG Shuqiang,TAN Shuang,et al. Hybrid range consistent hash partitioning strategy - a new data partition strategy for NoSQL database[C]//12th IEEE international conference on trust,security and privacy in computing and communications. Melbourne, VIC, Australia: IEEE, 2013:1161-1169.

[7] ITO J,SONOBE Y,IKEDA K,et al. Universal partitioning of the hierarchical fold network of 50-residue segments in proteins[J]. BMC Structural Biology,2009,9:34.

[8] 赵卫东,刘永红,鄢　涛,等. Oracle 分区表和分区索引在 VLDB 中的研究[J]. 成都大学学报:自然科学版,2016,35(4):358-360.

[9] 史　斌. 大型数据库分区表研究[J]. 中国新通信,2016,18(11):116-117.

[10] 卢朝霞,习　捷,王　剑. 数据库分区技术研究及应用[C]//2006 全国光电子与光电信息技术学术研讨会. 哈尔滨:出版者不详,2006.

[11] 庞　洁,闫　娟,韩鹏刚. 气井工程数据应用平台后台数据的查询优化[J]. 信息系统工程,2016(12):96.

[12] WANG Zhong. Design and application of large radar database[J]. Computer CD-ROM Software and Application, 2013,16(19):1-3.

[13] 胡廷波,钟　俊. 基于分簇的 B~+树数据库索引优化算法[J]. 计算机应用,2013,33(9):2472-2476.

[14] 谢　光. 数据库大数据量存储结构的探索[J]. 通讯世界,2017(11):29-30.

[15] 王　飞,谢小魁. Arcgis 二次开发综述[J]. 农业网络信息,2017(5):72-77.

[16] 徐继平. 基于 Linux 铸件数值模拟软件系统的研究于开发[D]. 武汉:华中科技大学,2006.

[17] 纪春华,石浩瀚,王艳磊. 基于 oracle 数据库整合技术的研究[J]. 信息技术与信息化,2018(11):90-96.

[18] 俞　海. QRACLE/MYSQL 数据库比较应用教学法综述[J]. 电脑知识与技术,2017,13(33):1-3.

[19] 郭月玲. 百分位数回归及应用研究[J]. 经济研究导刊,2013(22):9-10.

[20] 许璟龙. Oracle RAC 调优策略的研究与实现[D]. 大庆:东北石油大学,2012.