文章编号:1000-5641(2014)05-0301-10

面向 OLAP 应用的 OceanBase 模式设计

顾 伶[1], 翁海星[1], 胡华梁[1,2], 赵 琼[3]

(1. 华东师范大学 软件学院,上海 200062; 2. 浙江理工大学 经济管理学院,浙江 310018;
3. 交通银行 软件研发中心,上海 201201)

摘要: 随着大数据时代的来临,用户查询需求也越来越复杂,对数据库的可扩展性和 SQL 查询效率都提出了很高的要求. OceanBase 是 Alibaba 研发的适应互联网规模的关系型分布式数据库,能够做到可扩展、低成本、高可用,并能应用到很多业务场景,包括 OLTP 和 OLAP. 然而,当前发布的最新 OceanBase 只支持主键索引,还不能支持二级索引. 其次,OceanBase 在连接上没有并行处理,使得查询效率较低. 因此,为了能够使用主键索引及减少连接来加快查询速率,OceanBase 模式设计是必要的. 本文主要以 TPC-H 的 22 个查询为 OLAP 的研究案例,分析传统数据库的模式设计,并提出 OceanBase 分布式数据库下的模式设计规则,最后将会通过实验来验证这套模式设计规则的有效性.

关键词: OceanBase; OLAP; TPC-H

中图分类号: TP392 **文献标识码**: A **DOI**:10.3969/j.issn.1000-5641.2014.05.027

OceanBase schema design for OLAP application

GU Ling[1], WENG Hai-xing[1], HU Hua-liang[1,2], ZHAO Qiong[3]

(1. *Software Engineering Institute, East China Normal University, Shanghai* 200062, *China*;
2. *School of Economics and Management, Zhejiang Sci-Tech University, Hangzhou* 310018, *China*;
3. *Bank of Communications, Shanghai* 201201, *China*)

Abstract: As big data era is coming, high demands on the scalibility and query efficiency of database as user requirements are becoming more and more complicated. OceanBase developed by Alibaba is the relational distributed database. It is eqiupped with the feature of scalibility, low cost and availability. In addition, it is used in much wider applications, including OLTP and OLAP. However, the newest released version of OceanBase can only support the primary key index and cannot support the secondary index. Besides, OceanBase has no parallelism for join, which affects the query efficiency enormously. Therefore, the OceanBase schema design is necessary to make the primary key index and decreasing times of join useful. This paper studies TPC-H application as the OLAP example to analyse the relational database schema design and propose the OceanBase schema design. At last, we varify the efficiency of the schema design through experiments.

Key words: OceanBase; OLAP; TPC-H

收稿日期:2014-06

基金项目:浙江省自然科学基金(LY12F02044)

第一作者:顾伶,女,硕士研究生,研究方向为内存数据管理与分析. E-mail: guling@ecnu.edu.cn.

通信作者:胡华梁,男,副教授,硕士生导师,研究方向为数据库. E-mail: jmxyandy@126.com.

0 引 言

在当前的大数据时代,数据量已经达到了 PB 级,但是人们对于查询的要求越来越高,希望能在毫秒级时间内得到想要的结果. 传统的数据库存储方式过于简单,当数据量很大时,大量数据的堆积会导致服务器回应速率下降甚至崩溃,对企业造成很大的经济损失. 分布式数据库能够解决这样的问题,现在企业里常用的数据仓库有 Oracle、DB2、MySQL、Sybase、MS SQL Server 等,但是这些数据库都需要使用高性能的机器,并且扩展性有限,最多只能达到几百台机器,当前数据源多样,并且增长迅速,这样的数据库无法很好的提供大数据的存储以及查询服务. 随着经济迅速发展,方便大众的业务也在不断涌现. 例如,为了支持淘宝电子商务的业务,淘宝网和各家银行及物流公司都有合作. 这就需要淘宝的数据库中存储大量客户信息和交易信息,银行数据库中存储卡的交易信息,物流公司的数据库中存储客户的信息和邮件信息. 由于人口众多以及时代发展,无论是电子商务在线交易,还是其他业务的办理,都对数据库产生了极大压力.

OceanBase 是一个支持海量数据的高性能分布式数据库系统,能够实现数千亿条记录、数百 TB 数据上的跨行跨表事务. 这是由淘宝研发出来的适用于淘宝业务的数据库,具备高可用性和可靠性. OceanBase 能够承受住"双十一"在线交易的巨大压力,在 2013 年的"双十一"期间,事务量最大能达到 68000TPS,一天内的交易值达 17 亿笔之多. 使用 OceanBase 数据库,只要增加更多的机器,数据会自动迁移到新的机器上. 通过这样简单的扩展,淘宝成功渡过了交易量巨大的"双十一". 尽管 OceanBase 当前适用于淘宝,它处理大数据的能力却是金融行业所急需的.

OceanBase 也存在一些缺陷:未能支持 DATE、DECIMAL 等数据类型;也没有足够的查询优化:能够支持主键索引,未能支持 2 级索引. 金融企业的业务远比淘宝复杂,因此针对功能缺陷以及支持的索引来加快查询速率,OceanBase 的模式设计是非常必要的.

本文安排如下:第 1 节介绍 OceanBase 的整体框架、OLAP 查询的并行执行框架、单表和多表查询的执行计划以及 OceanBase 的功能缺陷. 第 2 节首先介绍了 TPC-H 的业务场景,并对其中的查询进行了分类,最后分析 OceanBase 对于单表和多表查询的模式设计. 第 3 节会通过实验来验证设计模式的有效性. 第 4 节总结全文.

1 OceanBase 功能简介

OceanBase 是一个既可以支持 OLAP 应用也可以支持 OLTP 应用的分布式关系型数据库,可以划分为 4 个模块[1]:主控服务器 RootServer、更新服务器 UpdateServer、基准数据服务器 ChunkServer 以及合并服务器 MergeServer. RootServer 实现对数据分布和数据副本的管理. UpdateServer 存储着增量更新数据,并且只有这个服务器允许写服务. ChunkServer 存储着 OceanBase 的基准数据. OceanBase 内部按照时间线将数据划分为基准数据和增量数据,基准数据是只读的,而所有的修改都会更新到增量数据中,系统内部会通过合并操作定期将增量数据融合到基准数据中. 当查询的时候,会先从 ChunkServer 取出数据,然后与 UpdateServer 上的增量数据进行合并,最后由 MergeServer 返回合并后的结果. 因此,为了减少查询时的合并操作,就需要在查询前进行增量数据和基准数据的合并. MergeServer 主要负责接受并解析用户的 SQL 请求,并且负责返回 ChunkServer 合并的结

果集. OceanBase 通过自己的容错机制来防止机器故障，以便在任何时候都能给用户提供服务.

OceanBase 对于 OLAP 应用能够实现并发查询功能，很大地提高了查询速率. 当用户进行查询时，MergeServer 首先进行 SQL 的解析，然后将大请求拆分为多个小请求，将小请求发送到有相应数据的 ChunkServer，然后由各个 ChunkServer 并发执行查询. OceanBase 会尽量实现 SQL 执行本地化，包括 Filter、Project、子请求部分的 GroupBy、OrderBy 等. 每个 ChunkServer 执行完查询，将结果返回给 MergeServer. 如果有 ChunkServer 上的查询执行失败，MergeServer 会将让存有副本的其他 ChunkServer 执行，当所有的查询都执行完，MergeServer 会对全部数据进行排序、分组等操作.

OceanBase 目前不支持的数据类型有 DECIMAL、DATE、CLOB、BLOB 和 NUMERIC，当要使用这几个类型的时候，可以根据具体场景用其他的类型替换. OceanBase 也不支持 EXISTS 的子句. 单表和多表查询的物理查询计划和其他数据库类似，但是 OceanBase 只是支持主键索引. 根据图 1 的单表查询的物理查询计划，由于 OceanBase 只支持主键索引，以对于 t1 表，可以将 c3 作为第一主键，加快了扫描的速度. 图 2 显示的是两表连接的一个查询. 对于连接的查询，on 后面的过滤项是没有索引作用的，所以需要将左边的 SQL 转换为中间的 SQL 形式. 对于左外连接的数据查询，OceanBase 中过滤条件位于 on 和 where 后面的过滤效果与传统数据库不一致，因而在实际业务中将其他数据库的左外连接的查询迁移到 OceanBase 中时，过滤条件的位置会影响结果的正确性. 例如下面 2 个 SQL，右表的条件写在 on 后面和 where 后面，当写在 on 后面的时候，数据库应该进行的操作是，先按照 B. c2 = 1 的条件进行了过滤，然后和 A 表进行连接，连接的结果是 B 表可以带着 NULL 的情况. 当条件写在 where 后面，就是 A 和 B 表做完 left join，然后将最后的结果根据 B. c2 = 1 进行过滤，这时候的效果与 INNER JOIN 相同. 另外，OceanBase 不支持非等值连接，也就是连接的 SQL 中一定要带有等值的连接.

(1) SELECT A. c1, B. c1 FROM A LEFT JOIN B ON A. c2 = B. c2 and B. c2 = 1;

(2) SELECT A. c1, B. c1 FROM A LEFT JOIN B ON A. c2 = B. c2 WHERE B. c2 = 1.

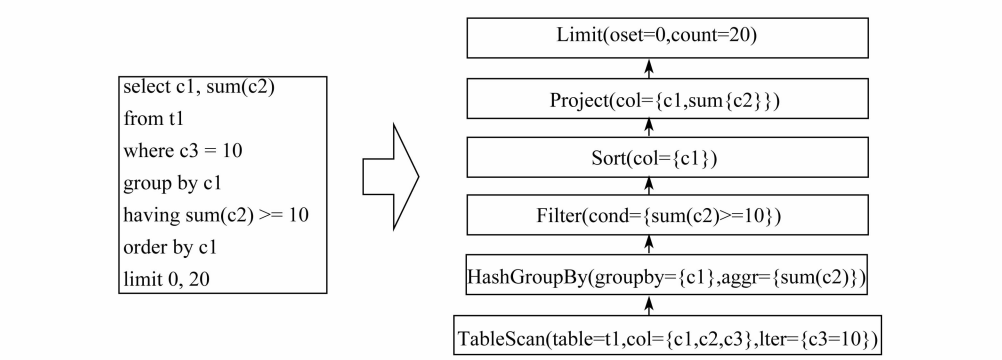

图 1　单表查询的物理查询计划

Fig. 1　Physical query plan of single table's query

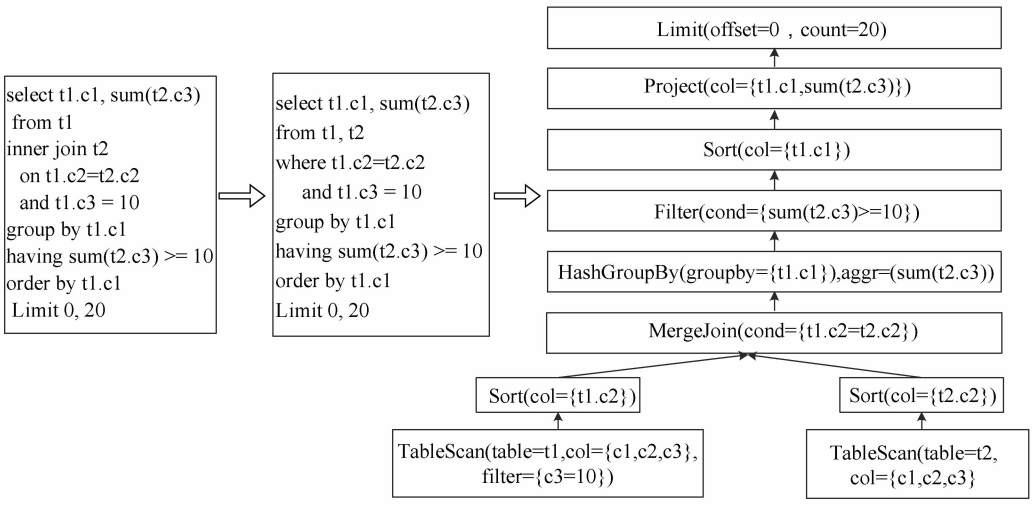

图 2 多表查询的物理查询计划

Fig. 2 Physical query plan of multiple tables' query

OceanBase 对 in 的使用有特别要求,以加快查询速率. 如果一张表有多个主键,那么当 in 语句中包含全部的主键的时候,这张表的查询会用上主键索引. 例如 A 表有主键 k1、k2,需要 WHERE (k1,k2) in (v1,v2). 最后,OceanBase 对 where 后面的子查询是不支持的,当出现这种情况,需要将子查询拆分出来,用中间查询结果集替换子查询.

2 OceanBase 下 TPC-H 案例研究

2.1 TPC-H

TPC Benchmark H 是一个决策支持的基准[5],它由一系列面向商务应用的查询和并行数据修改组成. 基准里选择的查询和组成数据库的数据在商业上都具有广泛的代表性并且易于实现. 目前 TPC-H 是 OLAP 领域常用的评测标准.

TPCH 一共有 8 张表:订单表(ORDERS),记录着每个客户的订单状态和总额;订单详情表(LINEITEM),记录着每个订单下的每个货品的状态;客户表(CUSTOMER),记录着客户信息;国家表(NATION),记录着国家的信息;地域表(REGION),记录着地域的信息;供应商表(SUPPLIER),记录着供应商的信息;供应货品表(PARTSUPP),记录着供应商提供的商品;货品表(PART),记录着货品信息. 表之间的关系如图 3 所示:

TPC-H 有 22 个复杂的查询,所选择的查询为各类商业分析提供定价和促销、供货和需求管理、利润和收入管理、顾客满意度研究、市场份额研究和运输管理. 根据 OceanBase 功能特点将 TPC-H 的查询进行了分类:

- 单表有复杂运算的查询:Q1,Q6
- 多表并且带有 exists 和 not exists 的查询:Q4,Q21,Q22
- 多表并且带有 not like 或者 not in 的查询:Q13,Q16
- where 或者 in 中有子句的查询:Q11,Q15,Q17,Q18,Q20
- 多表的有 or 条件:Q19

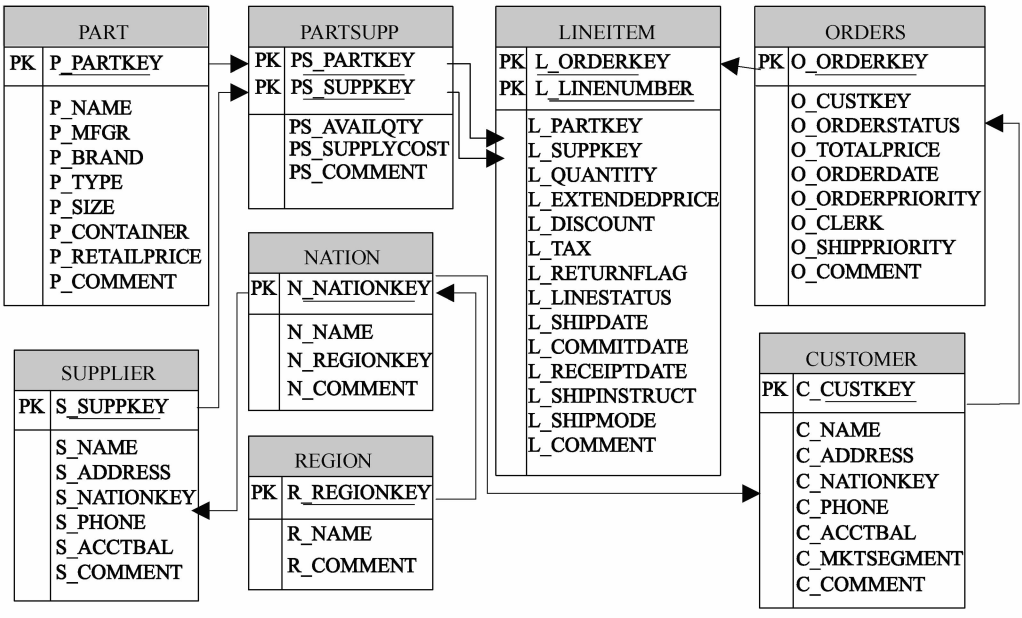

图 3　TPC-H 的模式

Fig. 3　The schema of TPC-H

• 简单的多表连接：Q2，Q3，Q5，Q7，Q8，Q9，Q10，Q12，Q14.

下面的章节会考虑 OceanBase 当前的架构，将此 TPC-H 的案例应用到 OceanBase，并且让每个查询的效率在 OceanBase 中达到最高，还会将 TPC-H 的查询分为单表查询和多表查询来进行研究.

2.2　单表查询的用例研究

OceanBase 只支持主键索引，若想加快单表查询的查询速率，就需要设置输入的属性为第 1 主键. 本节主要描述在单表查询下，TPC-H 模式的重新设计，使得单表查询的效率提高.

在实际生产中，OLTP 的数据在业务量少的时候会迁移到数据仓库中，为了保证 OLAP 的高效率查询，需要重新设计模式，并在数据传送时进行模式的变化. OceanBase 由 UpdateServer、ChunkServer 和 RootServer 组成，UpdateServer 存储的是增量数据，ChunkServer 存储的基准数据，RootServer 相当于 1 个 master 或者 1 个索引. 当进行查询时，OceanBase 会查询 ChunkServer 上的基准数据，然后与 UpdateServer 上的增量数据合并，最后返回给用户. 由于需要合并，查询速度受到了影响，所以在查询前需要对每天的数据进行合并，将 UpdateServer 上的数据合并到 ChunkServer 上.

由分类可以知道，单表查询只有 Q1 和 Q6. Q1 查询的是运送日期在 60～120 天内已经付款、已运送的和已返回的总金额. Q6 查询一年中在指定的百分比内的订单金额. 这两个查询都是使用了表 LINEITEM，但是表 LINEITEM 的主键是 L_ORDERKEY、LINENUMBER. OceanBase 支持主键索引，所以若使用原来的主键，则无法运用 OceanBase 的主键索引，查询速率低. 设置 LINEITEM 以 L_SHIPDATE 为第 1 主键，并且将原主键也作为主键以便确定唯一性，就能使用上主键索引，从而加快了查询速率.

2.3 多表查询的用例研究

该小节主要讨论多表连接的案例,统计了 TPC-H 中多表连接的案例并且列于下方. OceanBase 会将操作下推到各个服务器端,先对各个表进行过滤,然后将数据取到查询的 mergeserver 上,最后进行排序合并连接[2],因此需要尽量减少连接. 事实上,OceanBase 对连接没有优化,而且只是支持主键索引,所以需要在外部程序的帮助下,求出做连接的表的主键值,对每个表增加了主键索引,加快了查询速率. 首先列出 TPC-H 中连接的各种情况,然后分析对 TPC-H 的雪花状模型[3,4,6]的改变,以减少连接. 最后将以连接最多的 Q8 为案例,讲解在新的模型下的查询变化.

表 1 多表连接的分类情况

Tab. 1 Classification of joins

序号	查询编号	连接涉及表	连接的描述
1	Q2	partsupp,supplier,nation,region	给定区域和具体零件,选择哪个供应者订货
2	Q3,Q18	customer,orders,lineitem	查询价格排在前十位的客户信息
3	Q12	orders,lineitem	给定某货品状态,查询相关订单状态
4	Q5	customer,orders,lineitem,supplier,nation,region	给定国家和时间范围,查询这个国家的收入
5	Q7	supplier,lineitem,orders,customer,nation,nation	查询零件在两个国家之间的折扣
6	Q8	part, supplier, lineitem, orders, customer, nation,nation,region	零件类型在某个国家所占的市场份额
7	Q9	part,supplier,lineitem,partsupp,orders,nation	查询一个给定的产品在国家的利润
8	Q10	customer,orders,lineitem,nation	一个季度中有返回零件的客户对收入的影响
9	Q11	partsupp,supplier,nation	给定某供应商库存标志查询,查询相关零件在所有可得零件总价值中的百分比
10	Q13, Q22	customer, orders	查询客户的订单情况
11	Q14,Q17,Q19	lineitem, part	查询某个特殊产品的销售情况
12	Q16	partsupp, part	查询评价好的商品的数量
13	Q21	supplier,lineitem,orders,nation	查询不能及时货运所需零件的供应商

TPC-H 模式是一个雪花状模型,适合于维度分析,但是从表 1 中可以总结出来,查询基本上是指标分析,都会进行连接. 为了减少连接,需要根据这几种连接来合并表结构. 由于当前的雪花状模型,当进行连接的时候会涉及好几张表. 分析经常出现的表连接有:(1) LINEITEM 和 ORDERS 连接,例如 LINEITEM 存有供应商的序号,ORDERS 表存有顾客的序号,如果一个查询中需要供应商和顾客的信息,那么就需要将 LINEITEM,ORDERS,SUPPLIER,CUSTOMER 进行连接,涉及到的查询有 Q3,Q5,Q7,Q8,Q9,Q10,Q12,Q18,Q21;(2) SUPPLIER,NATION 和 REGION,例如查询某个区域的售货商情况,涉及的查询有 Q2,Q5,Q7,Q8,Q11,Q21;3、CUSTOMER,NATION 和 REGION,例如查询某个区域的客户情况,涉及的查询有 Q5,Q7,Q10,Q21.

如果需要查询供应商国家与顾客国家之间的货物情况,就需要 LINEITEM,ORDERS,SUPPLIER,CUSTOMER,PART,NATION,REGION 的连接,下面就以连接最多的查询 Q8 为例来看一下两表合并后的 SQL 的变化:

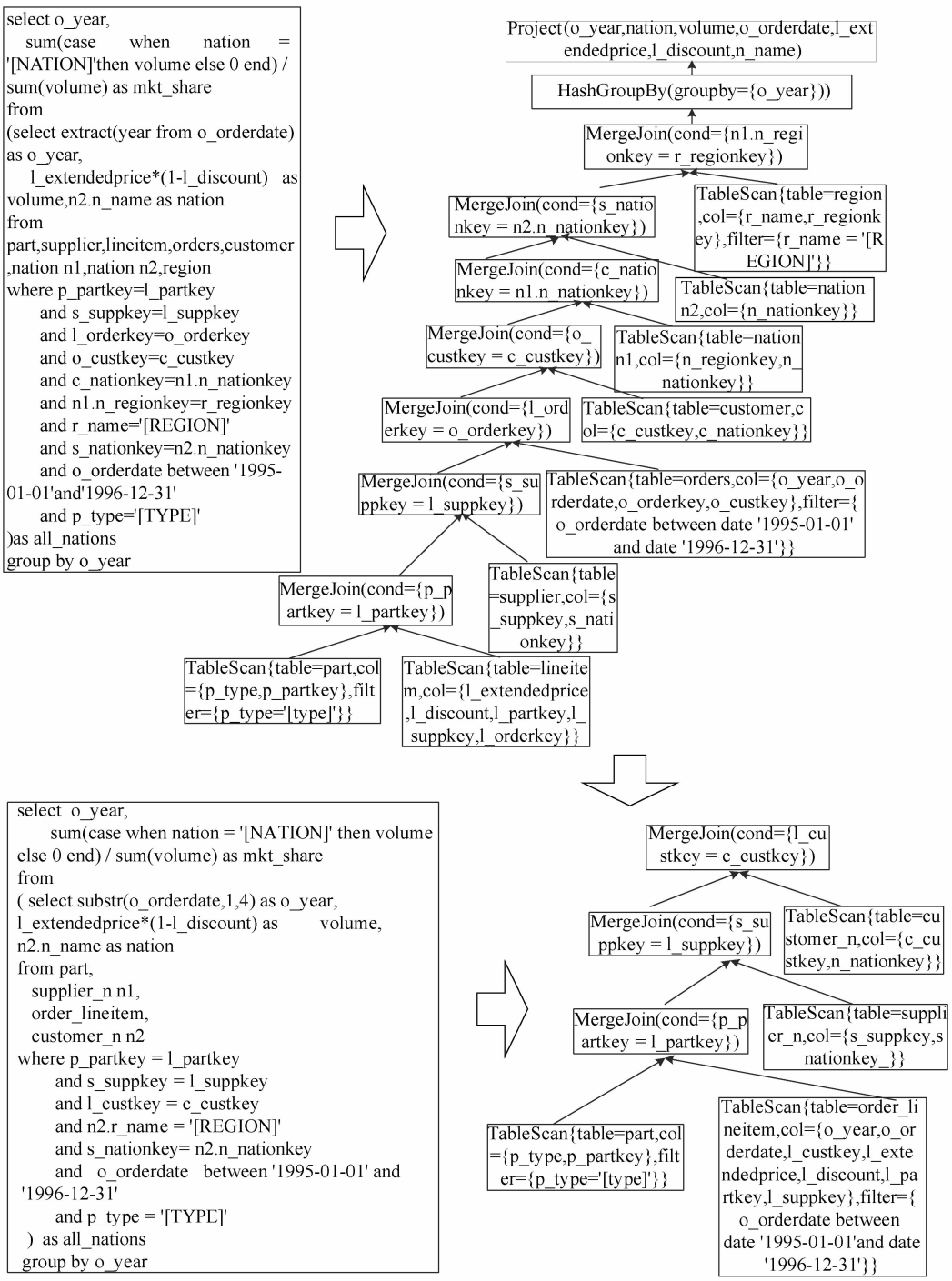

图 4　连接的案例

Fig. 4　The example of join

Q8 能够显示过去两年中一个给定的零件类型在某国某地区的市场份额的改变. 市场份额定义为某国某地区供应商供应特定种类的产品收入的百分比，是 l_extendedprice *（1-

ldiscount)的和.因为需要限定供应商和顾客的国家,时间需要用 ORDERS 的订单时间,这就需要将这么多的表进行连接.由查询看,只有 PART 表有 p_type 有限定条件,ORDERS 表有 o_orderdate 有限定.TPC-H 中最大的两个表为 ORDERS 和 LINEITEM,所以两表的连接用时最长,通过将 ORDERS 和 LINEITEM 两表合并为 ORDER_LINEITEM 的操作将两表查询改为单表查询.其次,只有 CUSTOMER 和 SUPPLIER 两表有 nation 和 region,所以可以让 CUSTOMER 和 NATION、REGION 通过 C_NATIONKEY = N_NATIONKEY,N_REGIONKEY = R_REGIONKEY 做连接,插入到新的表 CUSTOMER_N 中.表 SUPPLIER 和 NATION、REGION 也通过相同的方法插入到 SUPPLIER_N 中,这样就减少了三表连接.

例 1 显示了在新表下的查询,整个变化后的模式在图 5 中显示.

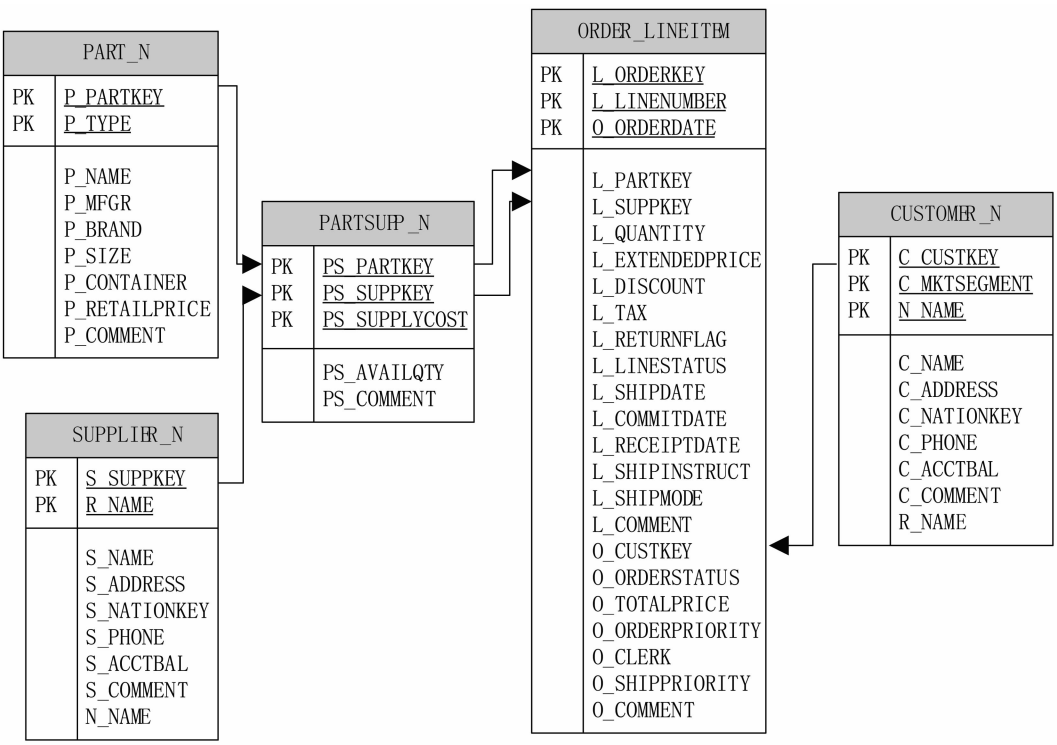

图 5 改变后的 TPC-H 的模式

Fig. 5 The altered schema of TPC-H

OceanBase 中对于 TPC-H 的模式与原来的模式不同之处有:将 CUSTOMER 和 NATION,REGION 合为 CUSTOMER_N;将 SUPPLIER 和 NATION,REGION 合为 SUPPLIER_N;将 LINEITEM 和 ORDERS 合为 ORDER_LINEITEM;新的模式中仍然保留 ORDERS,不保留 SUPPLIER 和 CUSTOMER,因为有些 SQL 会使用表 ORDERS,如果以 ORDER_LINEITEM 代替,那么扫描的数据量会增大;PART_N 的第 1 主键为 P_TYPE,SUPPLIER_N 的第 1 主键为 R_NAME,PARTSUPP_N 的第 1 主键为 PS_SUPPLYCOST,ORDER_LINEITEM 的第 1 主键为 O_ORDERDATE,CUSTOMER_N 的第 1 主键为 N_NAME.通过表结构的合并减少了表的连接,并且重新设置第 1 主键,从而提高了查

询速率.

3　实　　验

本节以 TPC-H 的前 20 个 SQL 为案例,根据上节所描述的对单表和多表的改写方式进行改写. 通过对比 OceanBase 上 TPC-H 的模式改变前后的查询效率,验证 OceanBase 上模式设计的有效性.

3.1　硬件配置及数据集

实验的硬件配置:

- CPU:Intel Xeon cpu e7-4870@2.4 GHZ 8 核
- 内存:50 G
- 硬盘:100 G
- 操作系统:Redhat 6.2
- 数据库:加上了 DECIMAL 运算的 OceanBase 0.4.2.8

实验所用的数据集:TPC-H 的数据生成器 DBgen,生成了 1 G 的数据,其中各个表的行数为:

- LINEITEM:6 001 215
- ORDERS:1 500 000
- CUSTOMER:150 000
- SUPPLIER:10 000
- PART:200 000
- PARTSUPP:800 000
- NATION:25
- REGION:5

3.2　模式改变前后效率对比

以 TPC-H 的前 20 个查询为案例,由于最后 2 个查询有 exists,所以未使用 Q21 和 Q22 查询. 首先,将 TPC-H 的 SQL 改为 OceanBase 能够支持的,使用原来的模式,记录在 OceanBase 中运行的总时间. 然后,将 TPC-H 的 SQL 改为新的模式下的 SQL,在 OceanBase 中运行后也记录下时间. 实验结果列于表 2,时间以秒为单位.

单表查询有 Q1 和 Q6,Q6 的效果很明显,因为 Q1 是对 LINEITEM 的查询,过滤项只是 L_SHIPDATE 小于某个时间,主键索引没有太大的作用,过滤出的数据集大于 Q6 的数据集,所以导致效果不是那么明显. 从 Q6 可以看出,在 OceanBase 中第 1 主键的改变能够使得查询速率提高 4 倍多.

多表连接的查询中有 14 个 SQL 运行时间是更少的. 剩余的 4 个多表查询运行时间长的原因是查询中没有直接对于 ORDER_LINEITEM 的过滤项,并且没有构建索引表. 对于多表查询,在没有构建索引表的情况下,大部分的 SQL 运行时间更短,因此表的合并使得查询效率提高.

从实验结果可以看出,在 OceanBase 中对于 TPC-H 模式的第 1 主键的改变以及表的合并使得查询效率提高了许多.

表 2 TPC-H 的查询时间对比表

Tab. 2 The running time comparison of TPC-H queries s

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
变化前 schema	176.98	19.51	61.80	200.71	76.57	64.55	136.52	147.16	251.05	84.38
变化后 schema	156.06	10.53	15.15	198.56	80.01	14.20	98.95	140.11	200.34	40.03
	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
变化前 schema	27.26	74.65	21.67	40.79	82.13	1415.71	104.98	107.53	194.78	52.84
变化后 schema	25.24	11.02	70.88	8.28	16.65	600.45	105.36	108.74	71.72	48.45

4 总 结

本文讲述了面向 OLAP 应用的 OceanBase 的模式设计,并将 TPC-H 作为案例研究,通过对比实验,说明了模式设计的有效性. OceanBase 支持主键索引,但是对于连接操作还没有并行处理机制,并且在左外连接上有缺陷,会影响结果的正确性. 根据 OceanBase 的特点,在模式设计的过程中,需要根据查询的过滤项来改变表的第 1 主键. 如果表的数据量小于 10 000 行,可以不需要主键索引. 其次,OceanBase 支持排序合并连接,所以 OceanBase 需要尽可能减少连接. 在模式设计过程中,可以根据业务场景来减少连接,将多个表合为 1 张表,将多表查询转换为单表查询,再通过第 1 主键的设置,可以加快查询速率.

[参 考 文 献]

[1] 杨传辉. 大规模分布式存储系统:原理解析与架构实战[M]. 北京: 机械工业出版社,2013.

[2] 莫利纳 H,厄尔曼 J,怀德姆 J. 数据库系统实现[M]. 杨冬青,吴愈青,包小源,等译. 2 版. 北京: 机械工业出版社,2010.

[3] 袁霖, 康慕宁, 李建良, 等. 一个面向 OLAP 应用的多维数据查询语言及其在对象关系数据库中的实现[J/OL]. 计算机工程与应用,2004,13: 182-218.

[4] IMHOFF C, NICHOLAS. Mastering data warehouse design: relational and dimensional techniques [M]. John Wiley & sons, 2004.

[5] Transaction Processing Performance Council (TPC). TPC BENCHMARKTM H [EB/OL]. (1993)[2013-01-02]. http://www.tpc.org/tpch/spec/tpch2.17.0.pdf.

[6] 王珊, 萨师煊. 数据库系统概论[M]. 4 版. 北京: 高等教育出版社,2006.

(责任编辑 赵 伟)