(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104850618 A
(43)申请公布日 2015.08.19

(21)申请号 201510250580.4

(22)申请日 2015.05.18

(71)申请人 北京京东尚科信息技术有限公司
地址 100080 北京市海淀区杏石口路65号西杉创意园四区11C楼东段1-4层西段1-4层
申请人 北京京东世纪贸易有限公司

(72)发明人 张成远 田琪 季锡强

(74)专利代理机构 中原信达知识产权代理有限责任公司 11219
代理人 张焕生 谢丽娜

(51)Int.Cl.
G06F 17/30(2006.01)

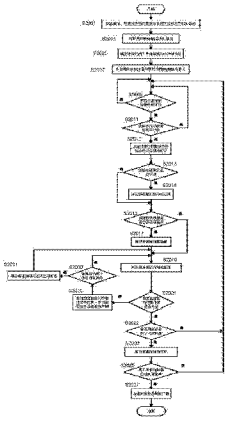
权利要求书2页 说明书7页 附图2页

(54)发明名称

一种提供有序数据的系统和方法

(57)摘要

本发明涉及一种提供有序数据的系统和方法，该系统包括：客户端，用于生成带有排序要求的查询请求；多个数据源，所述多个数据源的每个根据相应的查询子请求产生排序的数据记录并传递到对应的连接；连接部分，用于管理和分配到各数据源的连接；排序机构，用于通过相应连接处理来自各数据源的局部排序的数据记录以向客户端提供全局排序的数据记录。排序机构利用与各连接对应的缓冲区的界限标志来控制内存的使用，并且通过反复对各连接做堆排序来完成数据的全局排序。



CN 104850618 A

1.一种提供有序数据的方法，包括步骤：

从客户端接收带有排序要求的查询请求，并解析所述查询请求以产生对应于各数据源的子查询请求；

获取各数据源对应的连接，建立与各连接对应的缓冲区并清空且标记为未满，并且把产生的各子查询请求通过相应连接发送给相应的数据源，其中所述数据源通过相应连接返回有序的数据记录；

按照预定的规则来轮询各连接以确定哪个连接有数据可读取，其中，当确定一连接有数据可读取并且该连接对应的内存缓冲区未满，则读取该连接中所有可读的数据记录并存储到该连接对应的内存缓冲区中，当该缓冲区中的数据量超过预定阈值时把该缓冲区标记为已满，并且当确定需要通过该连接读取的数据全部读取完毕时把该连接标记为读取结束；

执行所有连接的堆排序；

当堆顶的连接对应的内存缓冲区非空时，从该内存缓冲区中取出第一个记录发送给客户端，并且，重复所有连接的堆排序并把堆顶的连接对应的内存缓冲区中的第一个记录取出发送给客户端，直到堆顶的连接对应的内存缓冲区为空，其中在从堆顶对应的内存缓冲区中取出第一个记录发送给客户端之后要判断该内存缓冲区是否未满，当未满时，取消该内存缓冲区的已满标记；

当堆顶的连接对应的内存缓冲区为空并且堆顶的该连接被标记读取结束，则标记该连接处理完毕，否则继续所述的轮询；

当所有的连接都已经处理完毕，则把结束标志发送给客户端，否则继续所述轮询，

其中，所述连接按照如下规则取值：

如果该连接被标记为处理完毕，对升序排序该连接取值无限大，对降序排序该连接的取值无限小；

如果该连接未被标记为处理完毕且该连接对应的缓冲区为空，对升序排序该连接的取值无限小，对降序排序该连接的取值无限大；

如果不是上述两种情况，该连接的值是该连接对应的缓冲区中第一个记录的排序字段值。

2.根据权利要求1所述的方法，所述数据源是MySQL实例。

3.根据权利要求1所述的方法，其中所述按照预定的规则来轮询各连接是根据设定的时间间隔来定期对各连接做轮询。

4.根据权利要求1所述的方法，所述连接是在TCP层实现的连接。

5.一种提供有序数据的系统，包括：

客户端，用于生成带有排序要求的查询请求；

多个数据源，所述多个数据源的每个根据相应的查询子请求产生排序的数据记录并传递到对应的连接；

连接部分，用于管理和分配到各数据源的连接；

排序机构，用于执行如下步骤：

从客户端接收带有排序要求的查询请求，并解析所述查询请求以产生对应于各数据源的子查询请求；

获取各数据源对应的连接，建立与各连接对应的缓冲区并清空且标记为未满，并且把产生的各子查询请求通过相应连接发送给相应的数据源，其中所述数据源通过相应连接返回有序的数据记录；

按照预定的规则来轮询各连接以确定哪个连接有数据可读取，其中，当确定一连接有数据可读取并且该连接对应的内存缓冲区未满，则读取该连接中所有可读的数据记录并存储到该连接对应的内存缓冲区中，当该缓冲区中的数据量超过预定阈值时把该缓冲区标记为已满，并且当确定需要通过该连接读取的数据全部读取完毕时把该连接标记为读取结束；

执行所有连接的堆排序；

当堆顶的连接对应的内存缓冲区非空时，从该内存缓冲区中取出第一个记录发送给客户端，并且，重复所有连接的堆排序并把堆顶的连接对应的内存缓冲区中的第一个记录取出发送给客户端，直到堆顶的连接对应的内存缓冲区为空，其中在从堆顶对应的内存缓冲区中取出第一个记录发送给客户端之后要判断该内存缓冲区是否未满，当未满时，取消该内存缓冲区的已满标记；

当堆顶的连接对应的内存缓冲区为空并且堆顶的该连接被标记读取结束，则标记该连接处理完毕，否则继续所述的轮询；

当所有的连接都已经处理完毕，则把结束标志发送给客户端，否则继续所述轮询，

其中，所述连接按照如下规则取值：

如果该连接被标记为处理完毕，对升序排序该连接取值无限大，对降序排序该连接的取值无限小；

如果该连接未被标记为处理完毕且该连接对应的缓冲区为空，对升序排序该连接的取值无限小，对降序排序该连接的取值无限大；

如果不是上述两种情况，该连接的值是该连接对应的缓冲区中第一个记录的排序字段值。

6. 根据权利要求 5 所述的系统，所述数据源是 MySQL 实例。

7. 根据权利要求 5 所述的系统，其中所述按照预定的规则来轮询各连接是根据设定的时间间隔来定期对各连接做轮询。

8. 根据权利要求 5 所述的系统，所述连接是在 TCP 层实现的连接。

一种提供有序数据的系统和方法

技术领域

[0001] 本发明涉及从多数据源获取有序数据的系统和方法。

背景技术

[0002] 随着互联网的不断发展，互联网上的数据量在急剧增长，传统单机数据库在处理大规模数据时已经面临明显的瓶颈，各大互联网公司都着手研究分布式数据库的实现方案。在分布式数据库的实现方案中包括两类，一类是客户端的解决方案，引入一个新的客户端，对数据进行分片处理，另一类是引入数据库中间件，对数据的分片处理由该中间件完成，应用程序只需要访问该数据库中间件即可，整个访问过程和访问原生的数据库几乎是一样的。

[0003] 在数据库中间件的解决方案中，针对 MySQL 的中间件的解决方案相对来说比较多，在实现 MySQL 中间件的时候需要解决的一个非常重要的问题就是从多个 MySQL 实例获取数据的时候如何对这些数据进行汇总排序。

[0004] 一些开源的数据库中间件如 Cobar 不支持涉及多个 MySQL 实例的数据排序。还有一些中间件支持排序，但是实现的时候是将所有涉及到的数据从各个 MySQL 实例上都获取到，然后将这些数据放在中间件所在机器的内存中或者是落到磁盘上，然后再对其进行排序，排完以后再发给客户端。

[0005] 中间件所在机器本身内存有限，在数据量大时就处理不了，如果需要将数据转到磁盘上则实现会比较复杂且性能会严重下降，而如果需要将数据专门汇总到另外的机器上，则会增加网络 IO，同样会降低性能。

发明内容

[0006] 本发明的目的是提供从多数据源有效获取有序数据的系统和方法。

[0007] 根据本发明的一个方面，提供一种提供有序数据的系统，包括：客户端，用于生成带有排序要求的查询请求；多个数据源，所述多个数据源的每个根据相应的查询子请求产生排序的数据记录并传递到对应的连接；连接部分，用于管理和分配到各数据源的连接；排序机构，用于执行如下步骤：从客户端接收带有排序要求的查询请求，并解析所述查询请求以产生对应于各数据源的子查询请求；获取各数据源对应的连接，建立与各连接对应的缓冲区并清空且标记为未满，并且把产生的各子查询请求通过相应连接发送给相应的数据源，其中所述数据源通过相应连接返回有序的数据记录；按照预定的规则来轮询各连接以确定哪个连接有数据可读取，其中，当确定一连接有数据可读取并且该连接对应的内存缓冲区未满，则读取该连接中所有可读的数据记录并存储到该连接对应的内存缓冲区中，当该缓冲区中的数据量超过预定阈值时把该缓冲区标记为已满，并且当确定需要通过该连接读取的数据全部读取完毕时把该连接标记为读取结束；执行所有连接的堆排序；当堆顶的连接对应的内存缓冲区非空时，从该内存缓冲区中取出第一个记录发送给客户端，并且，重复所有连接的堆排序并把堆顶的连接对应的内存缓冲区中的第一个记录取出发送给客户

端，直到堆顶的连接对应的内存缓冲区为空，其中在从堆顶对应的内存缓冲区中取出第一个记录发送给客户端之后要判断该内存缓冲区是否未满，当未满时，取消该内存缓冲区的已满标记；当堆顶的连接对应的内存缓冲区为空并且堆顶的该连接被标记读取结束，则标记该连接处理完毕，否则继续所述的轮询；当所有的连接都已经处理完毕，则把结束标志发送给客户端，否则继续所述轮询。其中，所述连接按照如下规则取值：如果该连接被标记为处理完毕，对升序排序该连接取值无限大，对降序排序该连接的取值无限小；如果该连接未被标记为处理完毕且该连接对应的缓冲区为空，对升序排序该连接的取值无限小，对降序排序该连接的取值无限大；如果不是上述两种情况，该连接的值是该连接对应的缓冲区中第一个记录的排序字段值。

[0008] 根据本发明的另一方面，提供一种提供有序数据的方法，包括步骤：从客户端接收带有排序要求的查询请求，并解析所述查询请求以产生对应于各数据源的子查询请求；获取各数据源对应的连接，建立与各连接对应的缓冲区并清空且标记为未满，并且把产生的各子查询请求通过相应连接发送给相应的数据源，其中所述数据源通过相应连接返回有序的数据记录；按照预定的规则来轮询各连接以确定哪个连接有数据可读取，其中，当确定一连接有数据可读取并且该连接对应的内存缓冲区未满，则读取该连接中所有可读的数据记录并存储到该连接对应的内存缓冲区中，当该缓冲区中的数据量超过预定阈值时把该缓冲区标记为已满，并且当确定需要通过该连接读取的数据全部读取完毕时把该连接标记为读取结束；执行所有连接的堆排序；当堆顶的连接对应的内存缓冲区非空时，从该内存缓冲区中取出第一个记录发送给客户端，并且，重复所有连接的堆排序并把堆顶的连接对应的内存缓冲区中的第一个记录取出发送给客户端，直到堆顶的连接对应的内存缓冲区为空，其中在从堆顶对应的内存缓冲区中取出第一个记录发送给客户端之后要判断该内存缓冲区是否未满，当未满时，取消该内存缓冲区的已满标记；当堆顶的连接对应的内存缓冲区为空并且堆顶的该连接被标记读取结束，则标记该连接处理完毕，否则继续所述的轮询；当所有的连接都已经处理完毕，则把结束标志发送给客户端，否则继续所述轮询。其中，所述连接按照如下规则取值：如果该连接被标记为处理完毕，对升序排序该连接取值无限大，对降序排序该连接的取值无限小；如果该连接未被标记为处理完毕且该连接对应的缓冲区为空，对升序排序该连接的取值无限小，对降序排序该连接的取值无限大；如果不是上述两种情况，该连接的值是该连接对应的缓冲区中第一个记录的排序字段值。

附图说明

[0009] 下面将参考附图详细地描述本发明的实施例，其中：

[0010] 图 1 是本发明的基于多数据源提供有序数据的系统组成以及数据排序的示意图；

[0011] 图 2 是本发明的基于多数据源提供有序数据的方法的流程图。

具体实施方式

[0012] 根据本发明的实施例，提供一种提供有序数据的系统，包括数据源部分、连接部分、排序机构部分及客户端。各数据源通过连接部分向排序机构提供局部有序的数据，经排序机构处理后，最终在客户端提供全局有序的数据。

[0013] 数据源部分的各数据源例如是 MySQL 实例。

[0014] 连接部分用于管理和分配到各数据源的连接，以实现排序机构对各数据源存取的管道，使得排序机构可从各数据源获取数据。

[0015] 排序机构从每个数据源获取的数据是按照规定顺序获取的，即各数据源提供的数据本身是局部有序的，例如都是递增顺序。排序机构需要对来自不同数据源的数据进行全局排序。

[0016] 排序机构例如是数据库中间件。该数据库中间件到一 MySQL 之间有一条 TCP 连接。MySQL 实例上的数据通过该连接不断发送给中间件。

[0017] TCP 连接的发送端和接收端各有一个缓冲区。MySQL 实例在向 TCP 连接发送数据的时候，首先会将数据放到 TCP 连接的发送端的缓冲区中，然后发送到相对的接收端，接收端接收到数据以后会先将数据存放到该 TCP 连接的接收端的缓冲区中，这些操作都是由操作系统本身完成的。

[0018] 中间件通过 TCP 连接接收 MySQL 实例上的数据，就是中间件读取 TCP 连接上的接收端的缓冲区中的内容。

[0019] 当接收端的缓冲区未满，MySQL 实例上的数据才能继续通过 TCP 连接发送过来，否则 MySQL 实例上的数据的发送将被阻塞。

[0020] 根据本发明的实施例，整个排序处理的示意图如图 1 所示，相应图上也有四个部分：

[0021] MySQL 实例集 101；

[0022] 中间件与 MySQL 实例之间的 TCP 连接 102；

[0023] 中间件排序机构 103；以及

[0024] 客户端 104。

[0025] 根据本发明的实施例，在图 1 中示出一种系统组成，其中数据源部分 MySQL 实例集 101 有六个 MySQL 实例。排序中间件 103 与每个 MySQL 实例有一条连接，共有六条连接 A、B、C、D、E 和 F。

[0026] 作为例子，假设每个 MySQL 实例上存在一些数据，这些数据通过连接部分传输给排序中间件。连接 A 要传输的数据有 0、6 和 12，连接 B 要传输的数据有 1、7 和 13，连接 C 要传输的数据有 8 和 20，连接 D 要传输的数据有 9、15 和 21，连接 E 要传输的数据有 4 和 10，连接 F 要传输的数据有 5、11 和 17。每条连接上要传输的数据都是经排序的，假设顺序是从小到大。

[0027] 根据本发明的提供有序数据的方法包括如下所述的步骤。

[0028] 排序机构从客户端接收带有排序要求的查询请求，并解析所述查询请求以产生对应于各数据源的子查询请求。

[0029] 然后，排序机构获取上述各数据源所对应的连接，建立与各连接对应的缓冲区并清空且标记为未满，并且把产生的各子查询请求通过相应连接发送给相应的数据源。所述各数据源响应对应的子查询在其数据库中检索出相应的数据记录，并通过其对应的连接返回按要求排序的数据记录。

[0030] 排序机构 103 按照预定的规则来轮询各连接以确定哪个连接有数据可读取。当排序机构确定某一连接有数据可读取并且该连接对应的内存缓冲区未满，则读取该连接中所有可读的数据记录并存储到该连接对应的内存缓冲区中。当该缓冲区中的数据量超过预定

阈值时，把该缓冲区标记为已满。然后，排序机构确定需要通过该连接读取的数据是否已处理完毕，并且当需要通过该连接读取的数据全部读取完毕时把该连接标记为读取结束。

[0031] 然后，排序机构 103 执行所有连接的堆排序。连接的排序值按照如下规则确定：如果该连接被标记为处理完毕，对升序排序该连接取值无限大，对降序排序该连接的取值无限小；如果该连接未被标记为处理完毕且该连接对应的缓冲区为空，对升序排序该连接的取值无限小，对降序排序该连接的取值无限大；如果不是上述两种情况，该连接的值是该连接对应的缓冲区中第一个记录的排序字段值。

[0032] 在一次堆排序之后，确定位于堆顶的连接对应的缓冲区是否为空。当堆顶的连接对应的内存缓冲区非空时，从该内存缓冲区中取出第一个记录发送给客户端 104，然后重复所有连接的堆排序以及把堆顶的连接对应的内存缓冲区中的第一个记录取出发送给客户端的处理，直到堆顶的连接对应的内存缓冲区为空，其中在从堆顶对应的内存缓冲区中取出第一个记录发送给客户端之后要判断该内存缓冲区是否未满，当未满时，取消该内存缓冲区的已满标记。

[0033] 当堆顶的连接对应的内存缓冲区为空并且堆顶的该连接被标记读取结束，则标记该连接为处理完毕，否则继续所述的轮询，以试图读取某个连接中的数据。

[0034] 当所有的连接都已经处理完毕，则把结束标志发送给客户端表示排序过程结束，否则继续所述轮询，以试图读取某个连接中的数据。

[0035] 下面结合图 2 描述本发明提供有序数据的方法的详细过程。

[0036] 排序中间件接收来自客户端的带有排序要求的查询请求，例如是 SQL 请求。

[0037] 在步骤 S2001 排序中间件接收该查询请求并解析该查询请求。如果该查询请求涉及多个 MySQL 实例，则对该查询请求做拆分，产生对应于各 MySQL 实例的子查询请求。

[0038] 如果查询请求仅涉及单个 MySQL 实例，则不需要分解查询请求，中间件只需要接收该单个 MySQL 实例发送过来的数据即可。

[0039] 在步骤 S2003 排序中间件获取有关各 MySQL 实例的连接，然后在步骤 S2005 建立与这些连接对应的缓冲区用于存储数据记录，并且把这些缓冲区清空且标记为未满。

[0040] 在步骤 S2007 排序中间件通过这些连接把产生的子 SQL 请求发送给相应的 MySQL 实例。

[0041] 各个 MySQL 实例接收相应的子 SQL 请求，根据子 SQL 请求检索数据，并相应对数据做排序，然后通过对应的连接将有序数据返回给排序中间件。

[0042] 在步骤 S2009 排序中间件按照预定的规则来轮询哪些连接有数据可以读取，如果没有连接有数据可读，则继续等待，并重新轮询是否有连接有数据可读。例如，可根据设定的时间间隔来定期对各连接做轮询，以判断连接是否有数据可读取。

[0043] 如何执行这种判断依赖于连接的具体实现方式。例如，在 TCP 层实现的连接中，TCP 连接的发送端和接收端各有一个缓冲区。数据库例如 MySQL 实例在向 TCP 连接发送数据的时候，首先会将数据放到 TCP 连接的发送端的缓冲区中，然后发送到相对的接收端，接收端接收到数据以后会先将数据存放到该 TCP 连接的接收端的缓冲区中，这些操作都是由相应操作系统完成的。

[0044] 如果在步骤 S2009 判断存在连接已经可从中读取数据，则在步骤 S2011 判断该连接对应的内存缓冲区是否已满。

[0045] 这里的缓冲区是否已满是指缓冲区中的数据量是否已经超过了预定的阈值。在读取一连接中的数据时，应该把该连接中可读取的数据全部读取完毕。因此在读取一个连接的数据的过程中，即使该连接对应的缓冲区已满，也要把可读取的数据读取完毕并存储在该缓冲区中（例如缓冲区实际有用来保存“溢出部分”的空间）。

[0046] 当在步骤 S2011 判定缓冲区未满，则排序中间件在步骤 S2012 读取该连接中所有可读的数据记录并存储到该连接对应的内存缓冲区中，然后在步骤 S2013 判断缓冲区是否已满，当已满时时，在步骤 S2014 把该缓冲区标记为已满。

[0047] 本发明通过对连接的缓冲区设置“满”标记的这种方式来控制内存的使用，实现了效率和可用性的平衡。

[0048] 接下来，在步骤 S2015 根据读取的数据判断需要通过该连接读取的数据是否已经全部读取完毕，即该连接对应的数据源 (MySQL 实例）应该提供的数据是否已经全部被读取完毕。该判断可以根据本领域已知的技术来实现。如果已经读取完毕，则在步骤 S2017 把该连接标记为读取结束。

[0049] 接下来，在步骤 S2019 对所有的连接做堆排序，或称对所有连接进行一次堆化处理。

[0050] 堆化处理是本发明中提升性能的关键，因为如果只是采用传统的归并算法的思路来处理整个排序过程会导致多个连接上的数据重复比较。关于本发明的“堆化处理”的细节，在后面专门描述。

[0051] 对所有的连接进行堆化处理以后，在步骤 S2021 判断处于堆顶的连接对应的内存缓冲区是否为空。

[0052] 如果为空，则说明还有连接的数据没有发送过来，再次重新判断哪些连接的数据已经可读。

[0053] 如果堆顶的连接对应的内存缓冲区中有数据记录，则缓冲区中的第一个数据记录就是尚未输出给客户端的数据记录中最小的记录（这是指升序的情况，如果是降序排序，则是最大的记录），在步骤 S2029 从堆顶的连接的内存缓冲区中取出第一个记录发送给客户端。这里所说的“取出记录”是指从内存缓冲区中读出该记录之后，把该记录从内存缓冲区中删除，这样，堆顶的连接的内存缓冲区的原来的第二个记录变成了第一个记录，意味着该连接的“取值”发生了变化。接着在步骤 S2030 再判断该连接的内存缓冲区是否是满的，当已不满，则在步骤 S2031 取消该连接缓冲区的已满标记。然后，再转步骤 S2019 对所有的连接做堆化处理，然后继续视堆顶连接的内存缓冲区是否有数据决定是否继续循环处理还是重新去接收数据。

[0054] 当在步骤 S2021 判定位于堆顶的连接对应的缓冲区为空，则转步骤 S2022 判断该连接是否被标记读取结束。如果没有读取结束，则转步骤 S2009，否则在步骤 S2023 标记该连接处理完毕。

[0055] 然后，在步骤 S2025 判断是否所有的连接都已经处理完毕。

[0056] 如果所有的连接都已经处理完毕，则转步骤 S2027 把结束标志发送给客户端。例如，结束标志对应 MySQL 协议中是指 EOF 包。

[0057] 如果还有连接没有处理完，则再转 S2009 重新判断哪些连接的数据已经可读。

[0058] 根据本发明，连接的缓冲区的限额保证了中间件在处理排序的时候所需要的内

存是有限且可控的，而堆化部分保证了比较次数是最少的确保整个排序的时间复杂度是O(nlgn)。

[0059] 堆化处理

[0060] 归并算法是一种已知的排序算法，用于把两个或者多个有序的序列合并成一个有序的序列。假设排序顺序是从小到大的，在涉及到n个(n是大于2的整数)有序序列的时候每次需要将所有的序列对比一次，获取到最小的元素，然后将该最小的元素从相应的有序序列中移除。假设这些有序序列分别是记为1...k...n，其中序列k的首个元素是最小的。在将序列k的首个元素移除以后，所有的有序序列的第一个元素又得比较一次，即比较n(n-1)/2。但此时其余n-1个有序序列彼此之间已经比较过了，只需要将序列k的新的首个元素与n-1个有序序列的首个元素进行比较即可，即再比较n-1次。实际上因为其余的n-1个有序序列的首个元素已经比较过了，如果之前的比较结果都已经有记录，则只需要将序列k的新的首个元素与其余的n-1个有序序列的首个元素中的最小的元素比较即可，所以最理想情况下只需要再比较1次即可获取到下一个最小的元素，但有可能序列k的新的首元素可能比其他的序列的首个元素都大，那么还是需要跟其余的n-1个有序序列再比较一次。但如果之前n-1个序列的比较结果都已经全部记录下来且以堆的形式记录下来，则只需要比较lgn次即可。

[0061] 罗伯特·弗洛伊德(Robert W.Floyd)和威廉姆斯(J.Williams)在1964年共同发明了著名的堆排序算法(Heap Sort)。堆排序产生一种有序的堆结构，是完全的二叉树。堆分为最大堆(或称大根堆、大顶堆)和最小堆(或称小根堆、小顶堆)。在最大堆中，每个节点的值都不大于其父节点的值。在最小堆中，非叶子节点的值小于其孩子节点的值。

[0062] 对连接做堆化处理就是对各连接做堆排序。本发明的堆化处理中产生的堆(二叉树)中的节点元素是连接而不是连接中的记录。

[0063] 对于从小到大排序的情况，采用最小堆，即堆顶的元素最小，并且按照如下规则确定连接的取值：

[0064] 如果该连接对应的缓冲区为空(没有数据)，即数据还没有从MySQL实例上发送过来，该连接的值取无限小(可认为是系统的最小值，即本系统不可能有数值比这个最小值更小)；

[0065] 如果该连接要传输的数据都已经全部被处理完毕，该连接的值取无限大(可认为是系统的最大值，即本系统不可能有数值比这个最大值更大)；

[0066] 如果不是上述两种情况，该连接的值是该连接对应的缓冲区中第一个记录的排序字段值。

[0067] 在最小堆的情况下，如果有连接还没有数据过来，那么该空的连接一定会被推到堆顶，如果有连接的数据已经全部处理完毕了，那么该连接已经会被推到堆的最底部。除去这两种情况以外，其他时候首条记录为最小的连接一定是位于堆的堆顶。

[0068] 本发明的堆化就是使作为堆中元素的连接都能满足双亲节点连接的首条记录的值小于孩子节点连接中的首条记录的值。

[0069] 如果要求按照从大到小排序，本发明可采用最大堆，并在比较的时候用调整的比较函数。

[0070] 说明书和附图所示的实施例仅用于解释和说明，而非限制本发明的范围，本发明

由权利要求书来限定。

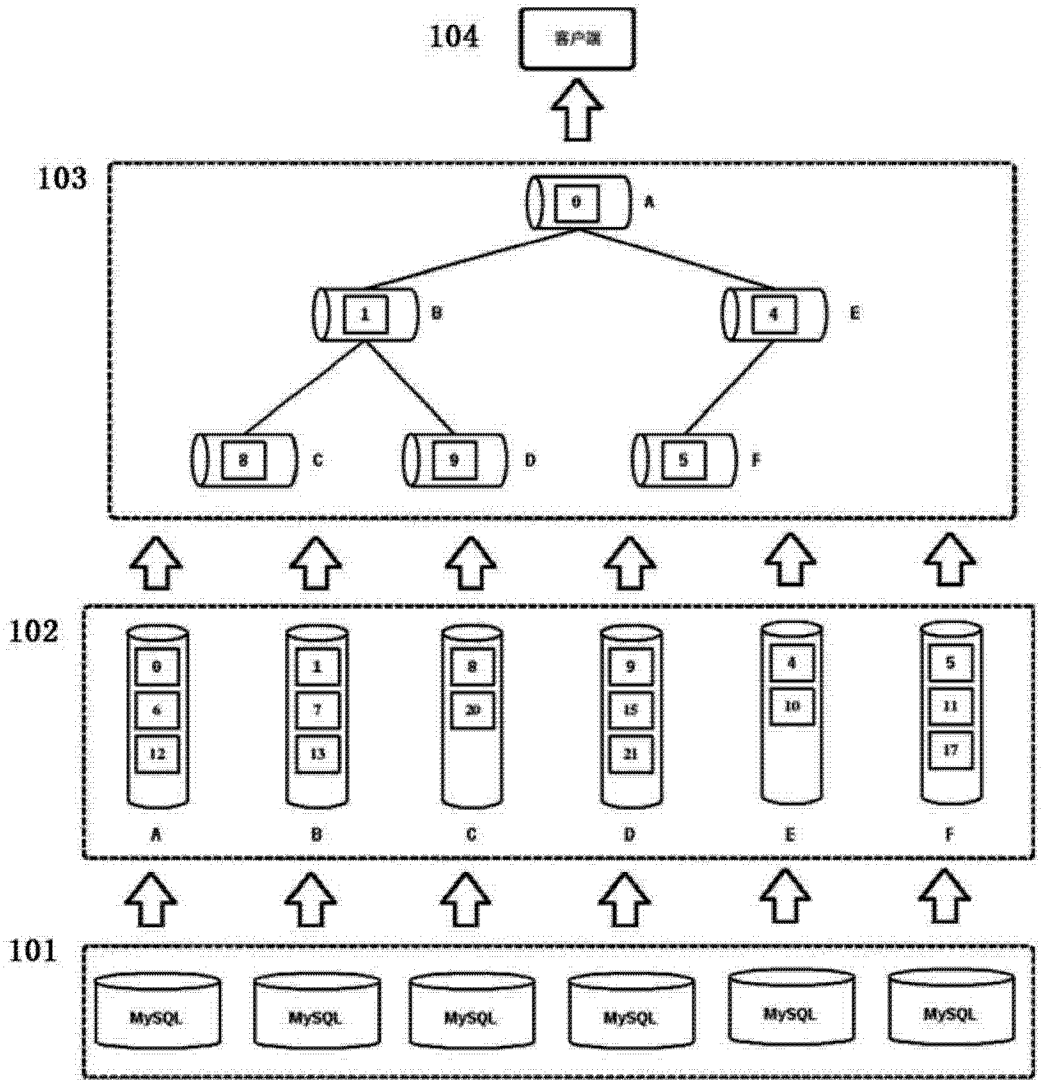

104
客户端
103
0
A
1
B
4
E
8
C
9
D
5
F
102
0
6
12
1
7
13
8
20
9
15
21
4
10
5
11
17
A
B
C
D
E
F
101
MySQL
MySQL
MySQL
MySQL
MySQL
MySQL

图 1

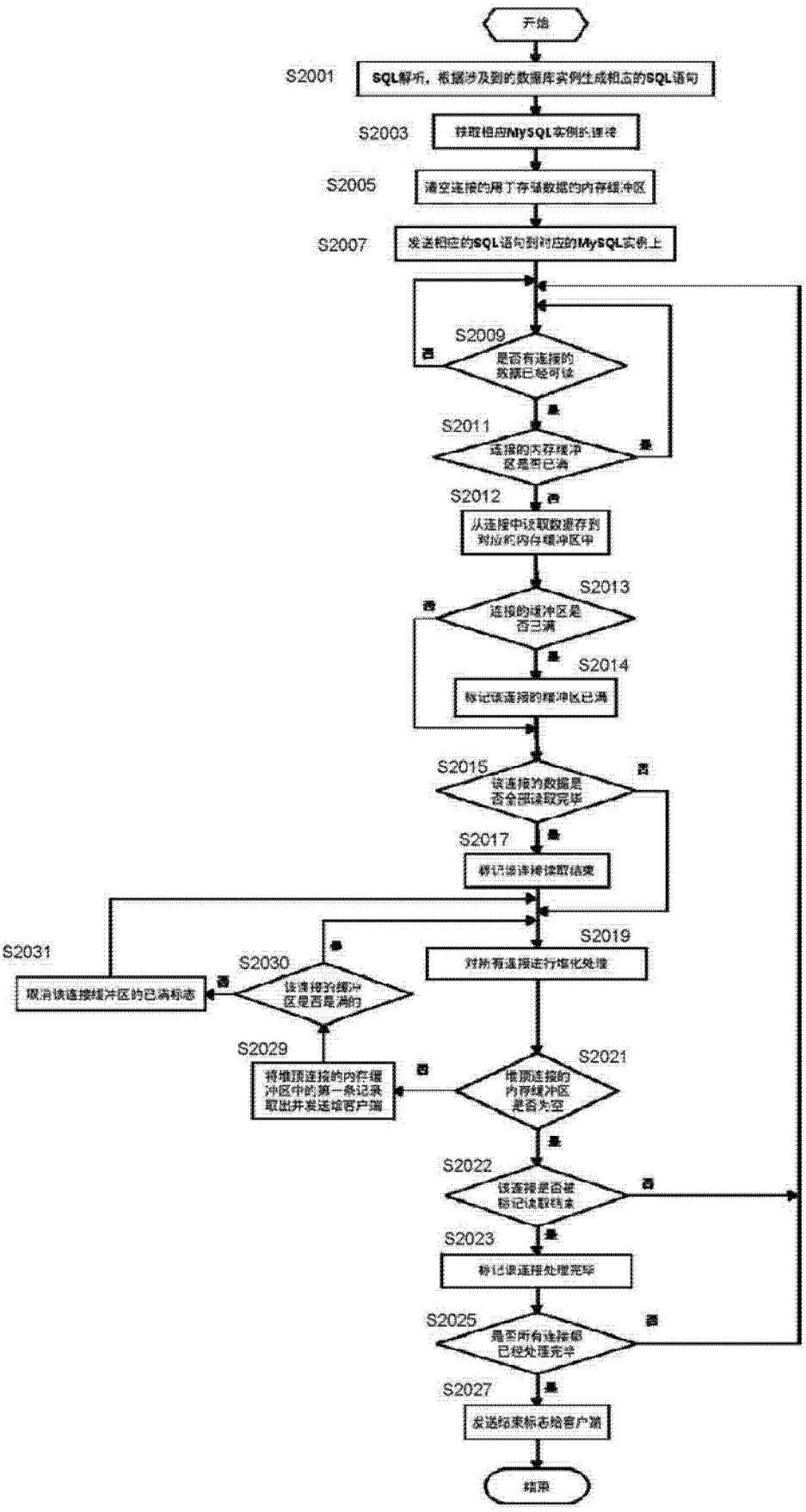

开始
S2001
SQL解析，根据涉及到的数据库实例生成相应的SQL语句
S2003
获取相应MySQL实例的连接
S2005
清空连接的用于存储数据的内存缓冲区
S2007
发送相应的SQL语句到对应的MySQL实例上
S2009
是否有连接的数据已经可读
否
是
S2011
连接的内存缓冲区是否已满
是
否
S2012
从连接中读取数据存到对应的内存缓冲区中
S2013
连接的缓冲区是否已满
否
是
S2014
标记该连接的缓冲区已满
S2015
该连接的数据是否全部读取完毕
否
是
S2017
标记该连接读取结束
S2019
对所有连接进行序化处理
S2021
堆顶连接的内存缓冲区是否为空
否
是
S2022
该连接是否被标记读取结束
否
是
S2023
标记该连接处理完毕
S2025
是否所有连接都已经处理完毕
否
是
S2027
发送结束标志给客户端
结束
S2029
将堆顶连接的内存缓冲区中的第一条记录取出并发送给客户端
S2030
该连接的缓冲区是否是满的
否
是
S2031
取消该连接缓冲区的已满标志

图 2