(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 November 2016 (17.11.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/183545 A1

(51) International Patent Classification:
G06F 12/02 (2006.01) *G06F 17/30* (2006.01)

(21) International Application Number:
PCT/US2016/032587

(22) International Filing Date:
14 May 2016 (14.05.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/161,813 14 May 2015 (14.05.2015) US

(71) Applicant: **WALLEYE SOFTWARE, LLC** [US/US]; 2800 Niagara Lane N., Plymouth, MN 55447 (US).

(72) Inventors: **TEODORESCU, Radu;** 303 East 109th Street, Apt 1, New York, NY 10029 (US). **CAUDY, Ryan;** 360 East 88th Street, Apt 25B, New York, NY 10128 (US). **KENT, David, R., IV;** 6965 Winter Hawk Circle, Colorado Springs, CO 80919 (US). **WRIGHT, Charles;** 48 Furnace Woods Road, Cortlandt Manor, NY 10567 (US). **RIES, Brian;** 2640 Xenwood Avenue South, St Louis Park, MN 55416 (US).

(74) Agent: **CARMICHAEL, James, T.;** Carmichael IP, PLLC, 8000 Towers Crescent Drive, 13th Floor, Tysons Corner, VA 22182 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available):* AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

[Continued on nextpage]

(54) Title: DISTRIBUTED AND OPTIMIZED GARBAGE COLLECTION OF REMOTE AND EXPORTED TABLE HANDLE LINKS TO UPDATE PROPAGATION GRAPH NODES

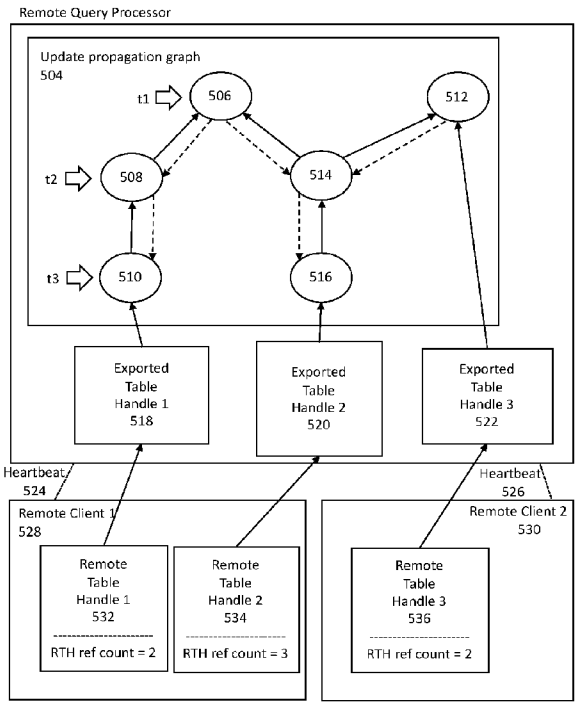

FIG. 5

(57) Abstract: Described are methods, systems and computer readable media for distributed and optimized garbage collection of remote and exported object handle links to update propagation graph nodes.

WO 2016/183545 A1

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available):* ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

DISTRIBUTED AND OPTIMIZED GARBAGE COLLECTION OF REMOTE AND EXPORTED TABLE HANDLE LINKS TO UPDATE PROPAGATION GRAPH NODES

[0001] This application claims the benefit of U.S. Provisional Application No. 62/161,813, entitled "Computer Data System" and filed on May 14, 2015, which is incorporated herein by reference in its entirety.

[0002] Embodiments relate generally to computer data systems, and more particularly, to methods, systems and computer readable media for providing optimized garbage collection of remote table handle links.

[0003] In an environment where a query server can have multiple remote clients, multiple links can be set up to tables on the query server from each remote client. There can be a system cost for maintaining all of these links, or from maintaining the data sources referenced by the links. As remote clients disconnect from from the query server or as remote clients stop using a particular table, links can still be registered for the remote clients on the query server. The number of links that need to be managed by the query server can continue to grow with new remote client connections and query operations against query server tables. If the links are not properly garbage collected after use, the amount of system resources to manage the increasing number of links and the data sources referenced by the links may also increase.

[0004] Embodiments were conceived in light of the above mentioned needs, problems and/or limitations, among other things.

[0005] Some implementations can include a system for managing distributed client-server object handles, the system comprising a remote client computer containing a first one or more hardware processors, a server computer containing a second one or more hardware processors, and the remote client computer containing a first computer readable data storage device coupled to the first one or more hardware processors, the first computer readable data storage device having stored thereon software instructions that, when executed by the first one or more hardware processors, cause the first one or more hardware processors to perfonn operations. The operations can include creating a remote object handle manager. The operations can also include establishing a connection with a remote query processor on the server computer. The operations can further include establishing a liveness indication system with the remote query processor. The operations can also include receiving from the remote query processor, exported object handle information for use in constructing a remote object

handle, including an exported object identifier, the exported object identifier identifying an exported object. The operations can include the remote object handle manager constructing a remote object handle. The operations can also include the remote object handle manager monitoring liveness of all client objects that depend on the remote object handle, the remote object handle depending on the exported object and indirectly on the exported object's dependencies. The operations can include the remote object handle manager sending a release notification to the remote query processor including an exported object identifier, after no client objects depend on the exported object. The operations can also include the server computer containing a second computer readable data storage device coupled to the second one or more hardware processors, the second computer readable data storage device having stored thereon software instructions that, when executed by the second one or more hardware processors, cause the second one or more hardware processors to perform operations. The operations can include creating a remote query processor, the remote query-processor performing operations.

[0006] The operations can include creating an exported object handle manager. The operations can also include sending the exported object handle information, including an exported object identifier from an exported object handle manager to the remote client computer. The operations can further include preserving a liveness of the exported object at least until the first of the following events: receipt of a release notification from the remote table handle manager; and the liveness indication system determines the remote client computer is not connected.

[0007] The remote query processor operations can include the remote query processor receiving a transmitted user query task from the remote client computer. The remote query processor operations can also include executing the transmitted user query task. The remote query processor operations can further include upon executing an iiistmction from the user query task to export an object, creating an exported object handle.

[0008] The remote query processor operations can include publishing a list of objects available for export to the remote client computer.

[0009] The operations can include wherein the remote object handle manager monitoring of client object liveness comprises maintaining a remote object handle reference count on the remote object handle. The operations can also include decrementing the remote object handle reference count after a dependent client object no longer depends on the remote object handle. The operations can further include when the remote object handle reference count

after decrementing is zero, sending a digital message to the remote **query** processor to release an associated exported object handle.

[0010] The operations can include wherein the exported object handle maintains a strong reference to one or more components of an update propagation graph created on the server computer.

[0011] The operations can include wherein **the** relationship between a remote object handle and its associated exported object extends an update propagation graph across **at** least one **of** multiple remote query processors and multiple clients.

[0012] The operations can include wherein **the** remote object handle invokes one or more methods on an exported object and delivers return values as copied objects or remote object handles of exported objects.

[0013] The operations can include wherein the remote client computer and the **server** computer are different computers.

[0014] The operations can include wherein the remote object handle manager monitoring of client object liveness comprises the remote object handle manager monitoring a handle cleanup reference queue and after a handle cleanup reference appears in the handle cleanup reference queue, invoking a handle cleanup reference cleanup method. The operations can further include the handle cleanup reference cleanup method removing the handle cleanup reference from a set of handle cleanup references monitored by the remote object handle manager, thereby eliminating **all** strong references to **the** handle cleanup reference. The operations can further include the handle cleanup reference cleanup method sending a digital message to the remote query processor **to** release an associated exported object handle.

[0015] The operations can include wherein the remote object handle manager monitoring of client object liveness comprises the remote object handle manager monitoring a handle cleanup reference queue and after a handle cleanup reference appears **in** the handle cleanup reference queue, invoking a handle cleanup reference cleanup method. The operations can also include the handle cleanup reference cleanup method decrementing a remote object handle reference count on a remote object handle associated with **the** handle cleanup reference. The operations can further include **the** handle cleanup reference cleanup method removing the handle cleanup reference from a set of handle cleanup references monitored by **the** remote object handle manager. The operations can include when **the** remote object handle reference count after decrementing is zero, sending a digital message to the remote **query** processor to release an associated exported object handle.

[0016] The operations can include wherein the remote query processor preserving a liveness of the exported object comprises maintaining a reference count associated with the exported object. The operations can also include decrementing the reference count associated with the exported object after receipt of a release notification from the remote table handle manager. The operations can further include decrementing the reference count associated with the exported object after the liveness indication system determines the remote client computer is not connected. The operations can also include when the remote object handle reference count after decrementing is zero, removing a strong reference to the exported object from the exported object handle. The operations can include when the remote object handle reference count after decrementing is greater than zero, maintaining a strong reference to the exported object from the exported object handle.

[0017] Some implementations can include a method for managing distributed client-server object handles, the method comprising creating a remote object handle manager. The method can also include establishing a connection with a remote query processor on a server computer. The method can further include establishing a liveness indication system with the remote query processor. The method can also include receiving from the remote query processor, exported object handle information for use in constructing a remote object handle, including an exported object identifier, the exported object identifier identifying an exported object. The method can include the remote object handle manager constructing a remote object handle. The method can also include the remote object handle manager monitoring liveness of all client objects that depend on the remote object handle, the remote object handle depending on the exported object and indirectly on the exported object's dependencies. The method can include the remote object handle manager sending a release notification to the remote query processor including an exported object identifier, after no client objects depend on the exported object. The method can also include creating a remote query processor, the remote query processor performing operations.

[0018] The operations can include creating an exported object handle manager. The operations can also include sending the exported object handle information, including an exported object identifier from an exported object handle manager to a remote client computer. The operations can further include preserving a liveness of the exported object until receipt of a release notification from the remote table handle manager or until the liveness indication system determines the remote client computer is not connected.

[0019] The remote query processor operations can further include the remote query processor receiving a transmitted user query task from a remote client computer. The operations can

also include executing the transmitted user query task. The operations can further include upon executing an instruction from the user query task to export an object, creating an exported object handle.

[0020] The remote query processor operations can include publishing a list of objects available for export to the remote client computer.

[0021] The method can include the operations of the first one or more hardware processors further comprising the remote object handle manager monitoring a handle cleanup reference queue. The operations can also include after a handle cleanup reference appears in the handle cleanup reference queue, invoking a handle cleanup reference cleanup method. The operations can further include the handle cleanup reference cleanup method decrementing a remote object handle reference count on a remote object handle associated with the handle cleanup reference. The operations can also include the handle cleanup reference cleanup method removing the handle cleanup reference from a set of handle cleanup references monitored by the remote object handle manager. The operations can include when the remote object handle reference count after decrementing is zero, sending a digital message to the remote query processor to release an associated exported object handle. The operations can also include when the remote object handle reference count after decrementing is greater than zero, maintaining a strong reference to the remote object handle in order to ensure iiveness for dependent client objects.

[0022] The method can include wherein the exported object handle maintains a strong reference to an update propagation graph created on the server computer.

[0023] The method can include wherein the relationship between a remote object handle and its associated exported object extends an update propagation graph across at least one of multiple remote query processors and multiple clients.

[0024] The method can include wherein the remote object handle invokes one or more methods on an exported object and delivers return values as copied objects or remote object handles of exported objects.

[0025] The method can include wherein the remote client computer and the server computer are different computers.

[0026] The method can also include wherein the preserving a Iiveness of the exported object comprises maintaining a reference count associated with the exported object. The method also includes decrementing the reference count associated with the exported object after receipt of a release notification from the remote table handle manager. The method further includes decrementing the reference count associated with the exported object after the

liveness indication system determines the remote client computer is not connected. The method also includes when the remote object handle reference count after decrementing is zero, removing a strong reference to the exported object from the exported object handle. The method further includes when the remote object handle reference count after decrementing is greater than zero, maintaining a strong reference to the exported object from the exported object handle.

[0027] Some implementations can include a nontransitory computer readable medium having stored thereon software instructions that, when executed by one or more processors, cause the one or more processors to perform operations. The operations can include creating a remote object handle manager. The operations can also include establishing a connection with a remote quen,' processor on a server computer. The operations can further include establishing a liveness indication system with the remote query processor. The operations can also include receiving from the remote query processor, exported object handle information for use in constructing a remote object handle, including an exported object identifier, the exported object identifier identifying an exported object. The operations can include the remote object handle manager constructing a remote object handle. The operations can also include the remote object handle manager monitoring liveness of all client objects that depend on the remote object handle, the remote object handle depending on the exported object and indirectly on the exported object's dependencies. The operations can include the remote object handle manager sending a release notification to the remote query processor including an exported object identifier, after no client objects depend on the exported object. The operations can also include the server computer containing a second computer readable data storage device coupled to the second one or more hardware processors, the second computer readable data storage device having stored thereon software instructions that, when executed by the second one or more hardware processors, cause the second one or more hardware processors to perform operations.

[0028] The operations can include creating a remote query processor, the remote query processor performing operations including creating an exported object handle manager. The remote quen,' processor operations can also include sending the exported object handle information, including an exported object identifier from an exported object handle manager to the remote client. The remote query operations can further include preserving a liveness of the exported object until receipt of a release notification from the remote table handle manager or until the liveness indication system determines the remote client is not connected.

[0029] The operations can include the remote query processor receiving a transmitted user query task from the remote client computer. The operations can also include executing the transmitted user query task. The operations can further include upon executing an instruction from the user query task to export an object, creating an exported object handle.

[0030] The operations can include publishing a list of objects available for export to the remote client.

[0031] The operations can include the remote object handle manager monitoring a handle cleanup reference queue. The operations can also include after a handle cleanup reference appears in the handle cleanup reference queue, invoking a handle cleanup reference cleanup method. The operations can further include the handle cleanup reference cleanup method decrementing a remote object handle reference count on a remote object handle associated with the handle cleanup reference. The operations can also include the handle cleanup reference cleanup method removing the handle cleanup reference from a set of handle cleanup references monitored by the remote object handle manager. The operations can include when the remote object handle reference count after decrementing is zero, sending a digital message to the remote query processor to release an associated exported object handle. The operations can also include when the remote object handle reference count after decrementing is greater than zero, maintaining a strong reference to the remote object handle in order to ensure liveness for dependent client objects.

[0032] The operations can include wherein the exported object handle maintains a strong reference to an update propagation graph created on the server computer.

[0033] The operations can include wherein the relationship between a remote object handle and its associated exported object extends an update propagation graph across at least one of multiple remote query processors and multiple clients.

[0034] The operations can include wherein the remote object handle invokes one or more methods on an exported object and delivers return values as copied objects or remote object handles of exported objects.

[0035] The operations can include wherein the remote client computer and the server computer are different computers.

[0036] Some implementations can include a method for monitoring liveness of parent objects and child objects participating in an update propagation graph on a query processing computer having parent listener objects and child listener objects, the method comprising maintaining continued liveness of said parent objects so long as their child objects have continued liveness. The method can also include permitting termination of liveness of

dependent child objects when their parent objects have continued iiveness. The method can further include maintaining continued iiveness of said parent listener objects so long as their child objects have continued Iiveness. The method can also include permitting termination of Iiveness of dependent child listener objects when their parent objects have continued Iiveness.

[0037] The method can include wherein the child objects are only weakly reachable from their parent objects and the parent objects are strongly reachable from their child objects.

BRIEF DESCRIPTION OF THE DRAWINGS

[0038] FIG. 1 is a diagram of an example computer data system showing an example data distribution configuration in accordance with some implementations.

[0039] FIG. 2 is a diagram of an example computer data system showing an example administration/process control arrangement in accordance with some implementations.

[0040] FIG. 3 is a diagram of an example computing device configured for remote query processor processing in accordance with some implementations.

[0041] FIG. 3A is a diagram of an example computing device configured for remote client processing in accordance with some implementations.

[0042] FIG. 4 is a diagram of an example update propagation graph in accordance with some implementations.

[0043] FIG. 4A is a diagram of an example update propagation graph with collected references in accordance with some implementations.

[0044] FIG. 5 is a diagram of an example remote query processor and remote clients in accordance with some implementations.

[0045] FIG. 5A is a diagram of an example remote table handle manager in accordance with some implementations.

[0046] FIG. 5B is a diagram of an example remote table handle manager in accordance with some implementations.

[0047] FIG. 5C is a diagram of an example remote table handle manager in accordance with some implementations.

[0048] FIG. 5D is a diagram of an example remote query processor and remote clients in accordance with some implementations.

[0049] FIG. 5E is a diagram of an example remote query processor and remote clients in accordance with some implementations.

[0050] FIG. 6 is a flowchart of an example remote table handle establishment in accordance with some implementations.

jOO5l] FIG. 7 is a flowchart of an example of loss of heartbeat reference collection in accordance with some implementations.

[0052] FIG. 8 is a flowchart of an example of non-loss of heartbeat reference collection in accordance with some implementations.

DETAILED DESCRIPTION

[0053] Reference is made herein to the Java programming language, Java classes, Java bytecode and the Java Virtual Machine (JVM) for puiposes of illustrating example implementations. It will be appreciated that implementations can include other programming languages (e.g., groovy, Scala, R, Go, etc.), other programming language structures as an alternative to or in addition to Java classes (e.g., other language classes, objects, data structures, program units, code portions, script portions, etc.), other types of bytecode, object code and/or executable code, and/or other virtual machines or hardware implemented machines configured to execute a data system query.

[0054] FIG. 1 is a diagram of an example computer data system and network 100 showing an example data distribution configuration in accordance with some implementations. In particular, the system 100 includes an application host 102, a periodic data import host 104, a query server host 106, a long-term file server 108, and a user data import host 110. While tables are used as an example data object in the description below, it will be appreciated that the data system described herein can also process other data objects such as mathematical objects (e.g., a singular value decomposition of values in a given range of one or more rows and columns of a table), TableMap objects, etc. A TableMap object provides the ability to lookup a Table by some key. This key represents a unique value (or unique tuple of values) from the columns aggregated on in a bvExternalQ statement execution, for example. A TableMap object can be the result of a byExternalQ statement executed as part of a query. It will also be appreciated that the configurations shown in FIGS. *I* and 2 are for illustration purposes and in a given implementation each data pool (or data store) may be directly attached or may be managed by a file server.

[0055] The application host 102 can include one or more application processes 112, one or more log files 114 (e.g., sequential, row-oriented log files), one or more data log tailers 116 and a multicast key-value publisher 118. The periodic data import host 104 can include a

local table data server, direct or remote connection to a periodic table data store 122 (e.g., a column-oriented table data store) and a data import server 120. The query server host 106 can include a multicast key-value subscriber 126, a performance table logger 128, local table data store 130 and one or more remote query processors (132, 134) each accessing one or more respective tables (136, 138). The long-term file server 108 can include a long-term data store 140. The user data import host 110 can include a remote user table server 142 and a user table data store 144. Row-oriented log files and column-oriented table data stores are discussed herein for illustration purposes and are not intended to be limiting. It will be appreciated that log files and/or data stores may be configured in other ways. In general, any data stores discussed herein could be configured in a manner suitable for a contemplated implementation.

[0056] In operation, the input data application process 112 can be configured to receive input data from a source (e.g., a securities trading data source), apply schema-specified, generated code to format the logged data as it's being prepared for output to the log file 114 and store the received data in the sequential, row-oriented log file 114 via an optional data logging process. In some implementations, the data logging process can include a daemon, or background process task, that is configured to log raw input data received from the application process 112 to the sequential, row-oriented log files on disk and/or a shared memory queue (e.g., for sending data to the multicast publisher 118). Logging raw input data to log files can additionally serve to provide a backup copy of data that can be used in the event that downstream processing of the input data is halted or interrupted or otherwise becomes unreliable.

[0057] A data log taiier 116 can be configured to access the sequential, row-oriented log file(s) 114 to retrieve input data logged by the data logging process. In some implementations, the data log taiier 116 can be configured to perform strict byte reading and transmission (e.g., to the data import server 120). The data import server 120 can be configured to store the input data into one or more corresponding data stores such as the periodic table data store 122 in a column-oriented configuration. The periodic table data store 122 can be used to store data that is being received within a time period (e.g., a minute, an hour, a day, etc.) and which may be later processed and stored in a data store of the long-term file server 108. For example, the periodic table data store 122 can include a plurality of data servers configured to store periodic securities trading data according to one or more characteristics of the data (e.g., a data value such as security symbol, the data source such as a given trading exchange, etc.).

[0058] The data import server 120 can be configured to receive and store data into the periodic table data store 122 in such a way as to provide a consistent data presentation to other parts of the system. Providing/ensuring consistent data in this context can include, for example, recording logged data to a disk or memory, ensuring rows presented externally are available for consistent reading (e.g., to help ensure that if the system has part of a record, the system has ail of the record without any errors), and preserving the order of records from a given data source. If data is presented to clients, such as a remote query processor (132, 134), then the data may be persisted in some fashion (e.g., written to disk).

[0059] The local table data server 124 can be configured to retrieve data stored in the periodic table data store 122 and provide the retrieved data to one or more remote query processors (132, 134) via an optional proxy.

[0060] The remote user table server (RUTS) 142 can include a centralized consistent data writer, as well as a data server that provides processors with consistent access to the data that it is responsible for managing. For example, users can provide input to the system by writing table data that is then consumed by query processors.

[0061] The remote query processors (132, 134) can use data from the data import server 120, local table data server 124 and/or from the long-term file server 108 to perform queries. The remote query processors (132, 134) can also receive data from the multicast key-value subscriber 126, which receives data from the multicast key-value publisher 118 in the application host 102. The performance table logger 128 can log performance information about each remote query processor and its respective queries into a local table data store 130. Further, the remote query processors can also read data from the RUTS, from local table data written by the performance logger, or from user table data read over NFS.

[0062] It will be appreciated that the configuration shown in FIG. 1 is a typical example configuration that may be somewhat idealized for illustration purposes. An actual configuration may include one or more of each server and/or host type. The hosts/servers shown in FIG. 1 (e.g., 102-110, 120, 124 and 142) may each be separate or two or more servers may be combined into one or more combined server systems. Data stores can include local/remote, shared/isolated and/or redundant. Any table data may flow through optional proxies indicated by an asterisk on certain connections to the remote query processors. Also, it will be appreciated that the term "periodic" is being used for illustration purposes and can include, but is not limited to, data that has been received within a given time period (e.g., millisecond, second, minute, hour, day, week, month, year, etc.) and which has not yet been stored to a long-term data store (e.g., 140).

[0063] FIG. 2 is a diagram of an example computer data system 200 showing an example administration/process control arrangement in accordance with some implementations. The system 200 includes a production client host 202, a controller host 204, a GUI host or workstation 206, and query server hosts 208 and 210. It will be appreciated that there may be one or more of each of 202-210 in a given implementation.

[0064] The production client host 202 can include a batch query application 212 (e.g., a query that is executed from a command line interface or the like) and a real time query data consumer process 214 (e.g., an application that connects to and listens to tables created from the execution of a separate query). The batch query application 212 and the real time query data consumer 214 can connect to a remote query dispatcher 222 and one or more remote query processors (224, 226) within the query server host *I* 208.

[0065] The controller host 204 can include a persistent query controller 216 configured to connect to a remote query dispatcher 232 and one or more remote query processors 228-230. In some implementations, the persistent query controller 216 can serve as the "primary client" for persistent queries and can request remote query processors from dispatchers, and send instructions to start persistent queries. For example, a user can submit a query to 216, and 216 starts and runs the query every day. In another example, a securities trading strategy could be a persistent query. The persistent query controller can start the trading strategy query every morning before the market open, for instance. It will be appreciated that 216 can work on times other than days. In some implementations, the controller may require its own clients to request that queries be started, stopped, etc. This can be done manually, or by scheduled (e.g., cron) jobs. Some implementations can include "advanced scheduling" (e.g., auto-start/stop/restart, time-based repeat, etc.) within the controller.

[0066] The GUI/host workstation can include a user console 218 and a user query application 220. The user console 218 can be configured to connect to the persistent query controller 216. The user query application 220 can be configured to connect to one or more remote query dispatchers (e.g., 232) and one or more remote query processors (228, 230).

[0067] FIG. 3 is a diagram of an example computing device 300 in accordance with at least one implementation. The computing device 300 includes one or more processors 302, operating system 304, computer readable medium 306 and network interface 308. The memory 306 can include remote query processor application 310 and a data section 312 (e.g., for storing AST's, precompiled code, etc.).

[0068] In operation, the processor 302 may execute the application 310 stored in the memory 306. The remote query processor application 310 can include software instructions that,

when executed by the processor, cause the processor to perform operations for distributed and optimized garbage collection of remote and exported table handle links to update propagation graph nodes in accordance with the present disclosure (e.g., performing one or more of 602-632; 702-708, 802-820 described below).

[0069] The remote query processor application program 310 can operate in conjunction with the data section 312 and the operating system 304.

[0070] FIG. 3A is a diagram of an example computing device 350 in accordance with at least one implementation. The computing device 350 includes one or more processors 352, operating system 354, computer readable medium 356 and network interface 358. The memory 356 can include remote client application 360 and a data section 362 (e.g., for storing ASTs, precompiled code, etc.).

[0071] In operation, the processor 352 may execute the application 360 stored in the memory 356. The remote client application 360 can include software instructions that, when executed by the processor, cause the processor to perform operations for distributed and optimized garbage collection of remote and exported table handle links to update propagation graph nodes in accordance with the present disclosure (e.g., performing one or more of 602-632; 702-708; 802-820 described below).

[0072] The remote client application program 360 can operate in conjunction with the data section 362 and the operating system 354.

[0073] Remote clients can have multiple connections to a query server host including communications to conduct query operations on the query server host that can result in creating an update propagation graph to map out static and dynamic nodes that represent the query execution. In order to allow a remote client to interact with nodes in the update propagation graph on the server, e.g. for display, remote method invocation, or querying, an exported table handle can be created on the server side and a remote table handle on the client side to handle the links between the remote client and the update propagation graph tables on the server.

[0074] FIG. 4 is a diagram of an example update propagation graph with collected references in accordance with some implementations. Variables (401, 403, 409, 411, 419) can be table or object reference variables. For example, variable t7 can refer to the result of a join operation on the referents of variable t8 and variable t9 (not shown). Variable VarO 401 can reference table object TO 402, variable Varl 403 can reference table object t1 404, variable Var2 409 can reference table object T2 410, Var3 411 can reference table object T3 412, and Var4 419 can reference table object T4 420.

[0075] Table object T2 410 can be a child node of table object TO 402 and table object T3 412 can be a child node of table object Tl 404. Table object T4 420 can be a child node of a join operation on table object T2 410 and table object T3 412. L0,2 406 can be a notification listener for propagating update notifications from table object TO to table object T2 410. LI, 3 408 can be a notification listener for propagating update notifications from table object Tl 404 to table object T3 412. L2,4 414 can be a notification listener for propagating update notifications from T2 410 to a merge join listener 4 418. L3,4 can be a notification listener for propagating update notifications from table object T3 also to merge join listener 4 418. Merge join listener 4 418 can be a notification listener for propagating join update notifications to table object T4 420. FIG. 4 contains two types of arrows, an arrow with a broken line and an arrow with a solid line. In this example, the arrows with broken lines represent weak references in the direction of the arrow for propagating update notifications from a parent, such as TO 402 to a child listener, such as LQ,2 406. Arrows with solid lines represent strong references in the direction of the arrow.

[0076] It will be appreciated that a weak reference does not prevent garbage collection of its referent, in this case a query update graph listener for propagating changes from a parent to a child node. A strong reference from a child node to a parent node serves to prevent garbage collection of the parent node until the child is no longer strongly reachable. Were the parent to hold strong references to the child listeners, this would prevent sub-graphs that are not externally reachable from being garbage collected.

[0077] FIG. 4A is a diagram of an example update propagation graph with collected references in accordance with some implementations. In this example, a query operation can re-assign a variable Var4 that previously referred to T4 420 to be NULL 422, thereby rendering T4 420 and its associated listeners Merge Join Listener 4 418, L2,4 414, and L3,4 416 only weakly reachable. Accordingly, the garbage collector can remove (make finalizable, finalize, and reclaim) table object T4 420 and listeners Merge Join Listener 4 418, L2,4 414, and L3,4 416 from memory and thus from the update propagation graph.

[0078] FIG. 5 is a diagram of an example remote query processor 502 and remote clients 524, 530 in accordance with some implementations. A remote query processor 502 can contain one or more update propagation graphs 504 and one or more exported table handles (518, 520, 522). An update propagation graph 504 can contain parent object nodes, such as 506, 512, and children nodes that are downstream to the parent object nodes, such as 508, 514, 510, 516. Some object nodes such 508, 514 can serve as both a parent and a child. For example, t1 table object 506 can be a parent to child t2 table object 508, and t2 table object

508 can be a parent to child t3 table object 510. The arrows with broken lines between the nodes can be weak references in the direction of the arrow and the arrows with solid lines between the nodes can be strong references. A remote client 528, 530 can have one or more remote table handles 532, 534, 536.

[0079] A remote table handle 532, 534, 536 can be created on a remote client 528, 530 to maintain an active link to a corresponding exported table handle 518, 520, 522 on a remote query processor 502. An exported table handle 518, 520, 522 can be created in response to a query operation on table objects in an update propagation graph 504. For example, Remote client 1 528 can send a query request in the form of t3=t2.select() to a remote query processor 502. In response, the remote query processor 502 can create table object t3 510 in the update propagation graph with the appropriate weak and strong references between t3 510 and t2 508. The remote query processor 502 can then create an exported table handle 1 518 with a strong reference to t3 510 and send a stub with the exported table handle information, including an identifier for the exported table to be used in subsequent messages, e.g. a handle release message, or a remote method invocation message, to the remote client 1 528. The remote client 1 528 remote table handle manager (not shown) can then create a remote table handle 1 532 to communicate with the exported table handle 518 on the remote query processor 502 for any further access to t3 510. Remote table handles 532, 534, 536 can contain a remote table handle (RTH) reference count. A remote table handle 532, 534, 536 can remain alive as long as its RTH reference count remains greater than zero.

[0080] It will be appreciated in the example that as long as the strong reference from the exported table handle 1 518 remains in place, t3 510 cannot be garbage collected. But if the strong reference from the exported table handle 1 518 is removed, t3 510 can then be garbage collected because the only reference to t3 510 would be a weak reference from the t3 510 parent, t2 508.

[0081] It will also be appreciated that the same exported table can be shared with multiple clients and that reference counting can occur on the exported table handle to track connected clients and to determine when to remove the exported table handles. It will also be appreciated that if the reference counting occurs on the exported table handle, then reference counting on the remote table handle may not be necessary.

[0082] A heartbeat signal 524, 526 can exist between each remote client 524, 526 and each remote query processor 502. A heartbeat signal 524, 526 can be used to determine whether a remote client 528, 530 is connected to a remote query processor 502 during period of remote client 528, 530 inactivity. It can be assumed by a remote query processor 502 that if a

heartbeat signal 524, 526 is no longer detected from a remote client 528, 530 that the remote client has disconnected from the remote query processor 502.

[0083] It will be appreciated that a heartbeat signal and a monitoring of a heartbeat signal is just one example of a liveness indication system. Other methods may exist for determining the status of connections between processes and/or computers.

[0084] It will be appreciated that one or more remote query processors 502 can be connected to one or more remote clients. One remote query processor 502 can be connected to one or more clients and one client can be connected to one or more remote query processors 502.

[0085] FIG. 5A is an example diagram of a remote client 1 528 that can contain a remote table handle manager 540. A remote table handle manager 540 can create and manage one or more remote table handles 532, 534, one or more handle cleanup references (HCR) (542, 544, 546, 548, 550) per each remote table handle 532, 534, one or more proxy objects (552, 556) per each HCR (542-550), one or more column objects (554, 558) per each HCR (542-550), one or more widgets 560 per HCR (542-550), and a cleanup queue 562.

[0086] A remote table handle 532, 534 can contain a remote table handle (RTH) reference count for each link to a handle cleanup reference (542-550). For example, remote table handle manager 540 can create an HCR for each proxy, column, and/or widget that is part of a table referenced by remote table handle 1 532. For example, if remote table handle 1 532 references a table with 1 proxy and 1 column, remote table handle manager 540 can create HCR1, 1 542 to reference the proxy 1 552 and HCR1, 2 to reference the column 1, 2 554. Continuing with the example, remote table handle 1 532 can have an RTH reference count of 2 because remote table handle *I* 532 is connected to 2 HCRs, HCR1, 1 542 and HCRl, 2 544. Each of the connections can be a weak reference as represented by an arrow with a broken line. Similarly, remote table handle 2 534 can have an RTH reference count of 3 because remote table handle 2 534 is connected to 3 HCRs, HCR2, 1 546, HCR2, 2 548 and HCR2, 3 550.

[0087] It will be appreciated that a table can have multiple proxies, columns and widgets. An HCR can be created for each proxy, column, and widget.

[0088] It will also be appreciated that a cleanup queue 562 can be empty or contain handle cleanup references that have been designated for cleanup. This concept is further described below.

[0089] FIG. 5B is an example diagram of a remote client 1 528 that can contain a remote table handle manager 540 with a widget reference removed from a remote table handle 534. In this example, a widget has been removed from the table represented by the remote table

handle 2 534. The system garbage collection code (not shown) can then remove the HCR2, 3 550 reference because the reference between the widget 2 560 and the HCR2, 3 550 is a weak reference as shown by the broken arrow, and a weak reference does not prevent garbage collection. The system garbage collection code can place the HCR2, 3 reference into the cleanup queue 562 because the link between the HCR2, 3 550 and widget 2 560 reference is a weak reference. The remote table handle manager 540 can monitor the cleanup queue 562. Next, the remote table handle manager 540 can dequeue HCR2, 3 from the cleanup queue 562 and can invoke an HCR cleanup method (not shown). The HCR cleanup method can then decrement the remote table handle 534 RTH reference count from 3 to 2 and remove HCR2, 3 from the set of HCRs being monitored by the remote table handle manager 540.

[0090] It will be appreciated that in this example, the remote table handle 2 532 remains alive because the RTH reference count for the remote table handle 2 532 remains at a value greater than zero.

[0091] It will also be appreciated that for the example where the reference counting occurs on the exported table handle instead of the remote table handle, then a separate exported table handle can be requested for each handle cleanup reference and dependent object on the client- It will also be appreciated that for the example of reference counting occurring on the exported table handle and not on the remote table handle, then cleanup references may not be used on the server side, except as an alternative to reference counting if a same object is exported multiple times. In this example, the server may not need to notify a client that an exported object is not reachable because the client may have already released the exported object.

[0092] FIG. 5C continues the example of FIG. 5B. A proxy 2 556 and a column 2, 2 558 can be removed from the table represented by the remote table handle 2 534. The system garbage collection code (not shown) can then remove the HCR2, 1 546 reference and the HCR2, 2 reference because the references are weak references as shown by the broken arrow, and a weak reference does not prevent garbage collection. The system garbage collection code can place the HCR2, 1 546 reference and the HCR2, 2 548 reference into the cleanup queue 562 because the link between the HCR2, *1* 546 reference and proxy 2 556, and the link between the HCR2, 2 548 reference and column 2, 2 558 are weak references. The remote table handle manager 540 can continue monitoring the cleanup queue 562. Next, the remote table handle manager 540 can dequeue HCR2, 1 and HCR2, 2 from the cleanup queue 562 and can invoke their HCR cleanup methods (not shown). The HCR cleanup methods can then decrement the remote table handle 534 RTH reference count from 2 to 0 and remove HCR2,

1 and HCR2,2 from the set of HCRs being monitored by the remote table handle manager 540. Because the remote table handle 2 534 now has a reference count of zero, the remote table handle manager 540 can remove the remote table handle 2 534 and take further action as shown in FIG. 5D to start a cleanup of matching resources on the server side.

[0093] FIG. 5D is a diagram of an example remote query processor and remote clients in accordance with some implementations. Continuing the example from FIG. 5C, the remote table handle manager 540 on remote client 1 528 can remove the remote table handle 2 534 because the RTH reference count for remote table handle 2 534 has been decremented to zero, thereby releasing client-side system sources that were used to maintain the remote table handle 2 534. The remote table handle manager 540 can also send a message to the remote query processor 502 to remove the exported table handle 2 520, thereby releasing server-side system resources that were used to maintain the exported table handle 2 520. The communication link between the remote table handle 2 534 and the exported table handle 2 520 can also be torn down, which can release communication resources for the link, such as sockets on both the client-side and server-side. Nodes 514 and 516 can then also be garbage collected because 514 and 516 loses the strong references to 514 and 516.

[0094] It will be appreciated that a connection can also exist per remote client - remote query processor relationship in place of or in addition to a remote table handle - exported table handle relationship.

[0095] FIG. 5E is a diagram of an example remote query processor and remote clients in accordance with some implementations. In this example, a heartbeat 526 signal between the remote client 2 530 and the remote query processor 502 has been lost. A remote query processor 502 can monitor the heartbeat 524, 526 between a remote query processor 502 and a remote client 528, 530. When a remote query processor 502 does not detect heartbeat 526 signal over a predetermined length of time, the remote client 530 can be determined to be either disconnecting or disconnected from the remote query processor 502. Steps can then be taken on both the client-side and the server side to cleanup resources that were freed up by the disconnection. For example, an exported table handle manager on the server side that is associated with the disconnected client can cleanup all of the exported table handles 522 associated with the disconnecting client 530. The strong references from the exported table handle to the update propagation graph 504 node 512 can also be removed. If the node 512 has no other strong references, the node can be made available for garbage collection. In this example, a strong reference still exists from node 514 to node 512, thus, node 512 can not be made available for garbage collection.

[0096] It will be appreciated that a single remote client can have many remote table handles connected to many exported table handles on the server-side, numbering into the hundreds and thousands. The cleanup of these connections after the loss of a heartbeat can release a significant amount of system resources.

[0097] FIG. 6 is a flowchart of an example remote table handle establishment in accordance with some implementations. Processing begins at 602, when a remote client establishes a connection with a remote query processor. Processing continues to 604.

[0098] At 604, the remote client and remote query processor establish a bidirectional heartbeat connection. Processing continues to 606.

[0099] At 606, the remote client transmits a user query to the remote query processor. Processing continues to 608.

[0100] At 608, the remote query processor executes the user query. Processing continues to 610.

[0101] At 610, the execution of the query may result in one or more strong references being removed from one or more nodes. Processing continues to 612.

[0102] At 612, when live nodes are added to an update propagation graph, the parent nodes hold weak references to their respective child listener objects, which in turn hold strong references to their parent table node, their child table node, and any other data stmctures they need for update propagation. The child table node holds a strong reference to its parent listeners, which hold strong references to the parent table object(s).

[0103] It will be appreciated that "live nodes" can be dynamic nodes that can listen for updates and pass on the update information to lower child nodes. Processing continues to 614.

[0104] At 614, the query causes changes in variables referencing update propagation graph nodes.

[0105] It will be appreciated that variables can be a holder for a strong reference to a table, such as "tl." The contents of a variable, such as tl, can change depending on how tl is used in a query instruction. For example, tl can initially be assigned as tl=t2.select() but can later be re-assigned as tl=t4. select. Processing continues to 616 and 618.

[0106] At 616, the remote query processor, during query execution, may explicitly create one or more export table handles. Processing continues to 624.

[0107] At 618, the remote query processor during execution of a query can publish the existence of tables that may be exported. Processing continues to 620.

[0108] At 620, client-side actions, either automatic or by user input, can request an exported table handle be created. Processing continues to 622.

[0109] At 622, the remote query processor creates an exported table handle on the remote query processor. Processing continues to 624.

[0110] At 624, the remote query processor sends a stub back to the remote client to use in constructing a remote table handle. Processing continues to 626.

[0111] At 626, a remote table handle is created on the remote client by a remote table handle manager. Processing continues to 628.

[0112] At 628, the remote table handle manager increments the remote table handle reference count and creates monitored handle cleanup references (HCR) for all objects that require the remote table handle and its corresponding exported table handle in order to remain active. Processing continues to 630.

[0113] At 630, the remote client establishes a link from the remote table handle to the exported table handle.

[0114] It will be appreciated that the link can be a logical link or a physical link. Processing continues to 632,

[0115] At 632, the remote query processor waits for the next query transmission from a remote client. Process returns to 606.

[0116] FIG. 7 is a flowchart of an example of loss of heartbeat reference collection 700 in accordance with some implementations. Processing begins at 702 when a remote query processor monitors for a heartbeat signal between the remote query processor and a remote client. Processing continues to 704.

[0117] At 704, a determination by the remote query processor is made as to whether a heartbeat signal the remote client is present. If the heartbeat is present, processing returns to 702 to continue monitoring for the heartbeat signal. If a heartbeat is not present, processing continues to 706.

[0118] At 706, an exported table handle manager on the remote query processor cleans up all of the exported table handles associated with the disconnecting or disconnected remote client. Processing continues to 708.

[0119] At 708, when no other strong links exist to the node that was connected to an exported table handle manager, the node is available for system initiated garbage collection.

[0120] FIG. 8 is a flowchart of an example of non-loss of heartbeat reference collection in accordance with some implementations. Processing begins at 802 and 804.

[0121] It will be appreciated that the two flows shown in FIG. 8 are shown without a direct connection between the two flows. The two flows in the example do not have a direct connection but do make use of a common queue, a handle cleanup reference (HCR) queue. One flow places objects into the HCR queue and the other flow retrieves the placed objects from the HCR queue. Steps of the first flow (802-810) are discussed first followed by step of the second flow (804-820).

[0122] At 802, a system garbage collector monitors object reachability, including the HCR's referent, which can be any object that requires the remote table handle and exported table handle to remain active. Processing continues to 806.

[0123] At 806, when the liveness-enforcing referent object of a handle cleanup reference becomes only weakly reachable, the HCR (as a weak reference) is cleared and the referent-object is marked finaiizable (eligible to be garbage collected by the system). Processing continues to 808.

[0124] At 808, the system garbage collection mechanism enqueues the HCR for the HCR queue. Processing continues to 810.

[0125] At 810, the system garbage collection mechanism finalizes and reclaims the liveness-enforcing referent that was previously weakly-reachable from the HCR.

[0126] At 804, a remote table handle manager monitors the HCR queue. Processing continues to 812.

[0127] At 812, the remote table handle manager dequeues an HCR and invokes an HCR cleanup method.

[0128] It will be appreciated that the term "method" here refers to a programming language constaict or function, and can be any constaict that is available in the programming language used to write the HCR cleanup function. Processing continues to 814.

[0129] At 814, the HCR cleanup method determines the remote table handle associated with the dequeued HCR and decrements that remote table handle reference count by one. Processing continues to 816.

[0130] At 816, the HCR cleanup method removes the HCR from the set of HCRs monitored by the remote table handle manager. Processing continues to 818.

[0131] At 818, the remote table handle manager determines whether the remote table handle reference count is zero. If the remote table handle count is greater than zero, processing returns to 804. If the remote table handle count is zero, processing continues to 820.

[0132] At 820, the remote table handle manager sends a digital message to the remote query processor to cleanup the associated exported table handle on the remote query processor. Processing returns to 804.

[0133] It will be appreciated that the modules, processes, systems, and sections described above can be implemented in hardware, hardware programmed by software, software instructions stored on a nontransitory computer readable medium or a combination of the above. A system as described above, for example, can include a processor configured to execute a sequence of programmed instructions stored on a nontransitory computer readable medium. For example, the processor can include, but not be limited to, a personal computer or workstation or other such computing system that includes a processor, microprocessor, microcontroller device, or is comprised of control logic including integrated circuits such as, for example, an Application Specific Integrated Circuit (ASIC), a field programmable gate array (FPGA), graphics processing unit (GPU), or the like. The instructions can be compiled from source code instructions provided in accordance with a programming language such as Java, C, C++, C#.net, assembly or the like. The instructions can also comprise code and data objects provided in accordance with, for example, the Visual Basic™ language, a specialized database query language, or another structured or object-oriented programming language. The sequence of programmed instructions, or programmable logic device configuration software, and data associated therewith can be stored in a nontransitory computer-readable medium such as a computer memory or storage device which may be any suitable memory apparatus, such as, but not limited to ROM, PROM, EEPROM, RAM, flash memory, disk drive and the like.

[0134] Furthermore, the modules, processes systems, and sections can be implemented as a single processor or as a distributed processor. Further, it should be appreciated that the steps mentioned above may be performed on a single or distributed processor (single and/or multi-core, or cloud computing system). Also, the processes, system components, modules, and sub-modules described in the various figures of and for embodiments above may be distributed across multiple computers or systems or may be co-located in a single processor or system. Example structural embodiment alternatives suitable for implementing the modules, sections, systems, means, or processes described herein are provided below.

[0135] The modules, processors or systems described above can be implemented as a programmed general purpose computer, an electronic device programmed with microcode, a hard-wired analog logic circuit, software stored on a computer-readable medium or signal, an optical computing device, a networked system of electronic and/or optical devices, a special

purpose computing device, an integrated circuit device, a semiconductor chip, and/or a software module or object stored on a computer-readable medium or signal, for example.

[0136] Embodiments of the method and system (or their sub-components or modules), may be implemented on a general-purpose computer, a special-purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit element, an ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discrete element circuit, a programmed logic circuit such as a PLD, PLA, FPGA, PAL, or the like. In general, any processor capable of implementing the functions or steps described herein can be used to implement embodiments of the method, system, or a computer program product (software program stored on a nontransitory computer readable medium).

[0137] Furthermore, embodiments of the disclosed method, system, and computer program product (or software instructions stored on a nontransitory computer readable medium) may be readily implemented, fully or partially, in software using, for example, object or object-oriented software development environments that provide portable source code that can be used on a variety of computer platforms. Alternatively, embodiments of the disclosed method, system, and computer program product can be implemented partially or fully in hardware using, for example, standard logic circuits or a VLSI design. Other hardware or software can be used to implement embodiments depending on the speed and/or efficiency requirements of the systems, the particular function, and/or particular software or hardware system, microprocessor, or microcomputer being utilized. Embodiments of the method, system, and computer program product can be implemented in hardware and/or software using any known or later developed systems or structures, devices and/or software by those of ordinary skill in the applicable art from the function description provided herein and with a general basic knowledge of the software engineering and computer networking arts.

[0138] Moreover, embodiments of the disclosed method, system, and computer readable media (or computer program product) can be implemented in software executed on a programmed general purpose computer, a special purpose computer, a microprocessor, or the like.

[0139] It is, therefore, apparent that there is provided, in accordance with the various embodiments disclosed herein, methods, systems and computer readable media for distributed and optimized garbage collection of remote and exported table handle links to update propagation graph nodes.

[0140] Application No. _____, entitled "DATA PARTITIONING AND ORDERING" (Attorney Docket No. W1.1-10057) and filed in the United States Patent and Trademark

Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0141] Application No. _____, entitled "COMPUTER DATA SYSTEM: DATA SOURCE REFRESHING USING AN UPDATE PROPAGATION GRAPH" (Attorney Docket No. W1.4-10058) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0142] Application No. _____, entitled "COMPUTER DATA SYSTEM: POSITION-INDEX MAPPING" (Attorney Docket No. Wl. 5-10083) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0143] Application No. _____, entitled "SYSTEM PERFORMANCE LOGGING OF COMPLEX REMOTE QUERY PROCESSOR QUERY OPERATIONS" (Attorney Docket No. W1.6-10074) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0144] Application No. _____, entitled "DISTRIBUTED AND OPTIMIZED GARBAGE COLLECTION OF REMOTE AND EXPORTED TABLE HANDLE LINKS TO UPDATE PROPAGATION GRAPH NODES" (Attorney Docket No. Wl. 8-10085) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0145] Application No. _____, entitled "COMPUTER DATA SYSTEM CURRENT ROW POSITION QUERY LANGUAGE CONSTRUCT AND ARRAY PROCESSING QUERY LANGUAGE CONSTRUCTS" (Attorney Docket No. W2. 1-10060) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0146] Application No. _____, entitled "PARSING AND COMPILING DATA SYSTEM QUERIES" (Attorney Docket No. W2.2-10062) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0147] Application No. _____, entitled "DYNAMIC FILTER PROCESSING" (Attorney Docket No. W2.4-10075) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0148] Application No. _____, entitled "DYNAMIC JOIN PROCESSING USING REAL-TIME MERGED NOTIFICATION LISTENER" (Attorney Docket No. W2.6- 10076)

and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0149] Application No. _____, entitled "DYNAMIC TABLE INDEX MAPPING" (Attorney Docket No. W2.7-10077) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0150] Application No. _____, entitled "QUERY TASK PROCESSING BASED ON MEMORY ALLOCATION AND PERFORMANCE CRITERIA" (Attorney Docket No. W2.8-10094) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0151] Application No. _____, entitled "A MEMORY-EFFICIENT COMPUTER SYSTEM: FOR DYNAMIC UPDATING OF JOIN PROCESSING" (Attorney Docket No. W2.9-10107) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0152] Application No. _____, entitled "QUERY DISPATCH AND EXECUTION ARCHITECTURE" (Attorney Docket No. W3.1-10061) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0153] Application No. _____, entitled "COMPUTER DATA DISTRIBUTION ARCHITECTURE" (Attorney Docket No. W3.2-10087) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0154] Application No. _____, entitled "DYNAMIC UPDATING OF QUERY RESULT DISPLAYS" (Attorney Docket No. W3.3-10059) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0155] Application No. _____, entitled "DYNAMIC CODE LOADING" (Attorney Docket No. W3.4-10065) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0156] Application No. _____, entitled "IMPORTATION, PRESENTATION, AND PERSISTENT STORAGE OF DATA" (Attorney Docket No. W3.5-10088) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0157] Application No. _____, entitled "COMPUTER DATA DISTRIBUTION ARCHITECTURE" (Attorney Docket No. W3.7-10079) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0158] Application No. _____, entitled "PERSISTENT QUERY DISPATCH AND EXECUTION ARCHITECTURE" (Attorney Docket No. W4.2- 10089) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0159] Application No. _____, entitled "SINGLE INPUT GRAPHICAL USER INTERFACE CONTROL ELEMENT AND METHOD" (Attorney Docket No. W4.3-10063) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0160] Application No. _____, entitled "GRAPHICAL USER INTERFACE DISPLAY EFFECTS FOR A COMPUTER DISPLAY SCREEN" (Attorney Docket No. W4.4-10090) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0161] Application No. _____, entitled "COMPUTER ASSISTED COMPLETION OF HYPERLINK COMMAND SEGMENTS" (Attorney Docket No. W4.5-10091) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0162] Application No. _____, entitled "HISTORICAL DATA REPLAY UTILIZING A COMPUTER SYSTEM" (Attorney Docket No. W5.1-10080) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0163] Application No. _____, entitled "DATA STORE ACCESS PERMISSION SYSTEM WITH INTERLEAVED APPLICATION OF DEFERRED ACCESS CONTROL FILTERS" (Attorney Docket No. W6.1-10081) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0164] Application No. _____, entitled "REMOTE DATA OBJECT PUBLISHING/SUBSCRIBING SYSTEM HAVING A MULTICAST KEY-VALUE PROTOCOL" (Attorney Docket No. W7.2-10064) and filed in the United States Patent and Trademark Office on May 14, 2016, is hereby incorporated by reference herein in its entirety as if fully set forth herein.

[0165] While the disclosed subject matter has been described in conjunction with a number of embodiments, it is evident that many alternatives, modifications and variations would be, or are, apparent to those of ordinary skill in the applicable arts. Accordingly, Applicants intend to embrace **all** such alternatives, modifications, equivalents and variations that are within the spirit and scope of the disclosed subject matter.

CLAIMS

What is claimed is:

1. A system for managing distributed client-server object handles, the system comprising:

a remote client computer containing a first one or more hardware processors;

a server computer containing a second one or more hardware processors;

the remote client computer containing a first computer readable data storage device coupled to the first one or more hardware processors, the first computer readable data storage device having stored thereon software instructions that, when executed by the first one or more hardware processors, cause the first one or more hardware processors to perform operations including:

creating a remote object handle manager;

establishing a connection with a remote query processor on the server computer;

establishing a liveness indication system with the remote query processor;

receiving from the remote query processor, exported object handle information for use in constructing a remote object handle, including an exported object identifier, the exported object identifier identifying an exported object;

the remote object handle manager constructing a remote object handle;

the remote object handle manager monitoring liveness of ail client objects that depend on the remote object handle, the remote object handle depending on the exported object and indirectly on the exported object's dependencies;

the remote object handle manager sending a release notification to the remote query processor including an exported object identifier, after no client objects depend on the exported object;

the server computer containing a second computer readable data storage device coupled to the second one or more hardware processors, the second computer readable data storage device having stored thereon software instructions that, when executed by the second one or more hardware processors, cause the second one or more hardware processors to perform operations including:

creating a remote query processor, the remote query processor performing operations including:

creating an exported object handle manager;

sending the exported object handle information, including an exported object identifier from an exported object handle manager to the remote client computer; and

preserving a liveness of the exported object at least until the first of the following events:

receipt of a release notification from the remote table handle manager; and

the liveness indication system determines the remote client computer is not connected.

2. The system of claim 1, the remote query processor operations further comprising:

the remote query processor receiving a transmitted user query task from the remote client computer;

executing the transmitted user query task, and

upon executing an instruction from the user query task to export an object, creating an exported object handle.

3. The system of claim 1, the remote query processor operations further comprising publishing a list of objects available for export to the remote client computer.

4. The system of claim 1, wherein the remote object handle manager monitoring of client object liveness comprises:

maintaining a remote object handle reference count on the remote object handle,

decrementing the remote object handle reference count after a dependent client object no longer depends on the remote object handle, and

when the remote object handle reference count after decrementing is zero, sending a digital message to the remote query processor to release an associated exported object handle.

5. The system of claim 1, wherein the exported object handle maintains a strong reference to one or more components of an update propagation graph created on the server computer.

6. The system of claim 1, wherein the relationship between a remote object handle and its associated exported object extends an update propagation graph across at least one of multiple remote query processors and multiple clients.

7. The system of claim 1, wherein the remote object handle invokes one or more methods on an exported object and delivers return values as copied objects or remote object handles of exported objects.

8. The system of claim 1, wherein the remote client computer and the server computer are different computers.

9. A method for managing distributed client-server object handles, the method comprising:

creating a remote object handle manager;

establishing a connection with a remote query processor on a server computer;

establishing a liveness indication system with the remote query processor;

receiving from the remote query processor, exported object handle information for use in constructing a remote object handle, including an exported object identifier, the exported object identifier identifying an exported object;

the remote object handle manager constructing a remote object handle;

the remote object handle manager monitoring liveness of all client objects that depend on the remote object handle, the remote object handle depending on the exported object and indirectly on the exported object's dependencies;

the remote object handle manager sending a release notification to the remote query processor including an exported object identifier, after no client objects depend on the exported object;

creating a remote query processor, the remote query processor performing operations including:

creating an exported object handle manager;

sending the exported object handle information, including an exported object identifier from an exported object handle manager to a remote client computer, and

preserving a liveness of the exported object until receipt of a release notification from the remote table handle manager or until the liveness indication system determines the remote client computer is not connected.

10. The method of claim 9, the remote query processor operations further comprising:

the remote query processor receiving a transmitted user query task from a remote client computer;

executing the transmitted user query task; and

upon executing an instruction from the user query task to export an object, creating an exported object handle.

11. The method of claim 9, the remote query processor operations further comprising publishing a list of objects available for export to the remote client computer.

12. The method of claim 9, the operations of the first one or more hardware processors further comprising:

the remote object handle manager monitoring a handle cleanup reference queue;

after a handle cleanup reference appears in the handle cleanup reference queue, invoking a handle cleanup reference cleanup method;

the handle cleanup reference cleanup method decrementing a remote object handle reference count on a remote object handle associated with the handle cleanup reference;

the handle cleanup reference cleanup method removing the handle cleanup reference from a set of handle cleanup references monitored by the remote object handle manager;

when the remote object handle reference count after decrementing is zero, sending a digital message to the remote query processor to release an associated exported object handle; and

when the remote object handle reference count after decrementing is greater than zero, maintaining a strong reference to the remote object handle in order to ensure iiveness for dependent client objects.

13. The method of claim 9, wherein the exported object handle maintains a strong reference to an update propagation graph created on the server computer.

14. The method of claim 9, wherein the relationship between a remote object handle and its associated exported object extends an update propagation graph across at least one of multiple remote query processors and multiple clients.

15. The method of claim 9, wherein the remote object handle invokes one or more methods on an exported object and delivers return values as copied objects or remote object handles of exported objects.

16. The method of claim 9, wherein the remote client computer and the server computer are different computers.

17. A nontransitory computer readable medium having stored thereon software instructions that, when executed by one or more processors, cause the one or more processors to perform operations including:

creating a remote object handle manager,

establishing a connection with a remote query processor on a server computer;

establishing a liveness indication system with the remote query processor;

receiving from the remote query processor, exported object handle information for use in constructing a remote object handle, including an exported object identifier, the exported object identifier identifying an exported object;

the remote object handle manager constructing a remote object handle,

the remote object handle manager monitoring liveness of all client objects that depend on the remote object handle, the remote object handle depending on the exported object and indirectly on the exported object's dependencies;

the remote object handle manager sending a release notification to the remote query processor including an exported object identifier, after no client objects depend on the exported object;

the server computer containing a second computer readable data storage device coupled to the second one or more hardware processors, the second computer readable data storage device having stored thereon software instructions that, when executed by the second one or more hardware processors, cause the second one or more hardware processors to perform operations including:

creating a remote query processor, the remote query processor performing operations including:

creating an exported object handle manager,

sending the exported object handle information, including an exported object identifier from an exported object handle manager to the remote client; and

preserving a liveness of the exported object until receipt of a release notification from the remote table handle manager or until the liveness indication system determines the remote client is not connected.

18. The nontransitory computer readable medium of claim 17, further comprising:

the remote query processor receiving a transmitted user query task from the remote client computer,

executing the transmitted user query task; and

upon executing an instruction from the user query task to export an object, creating an exported object handle.

19. The iiontransitory computer readable medium of claim 17, further comprising publishing a list of objects available for export to the remote client.

20. The nontransitory computer readable medium of claim 17, further comprising:

the remote object handle manager monitoring a handle cleanup reference queue;

after a handle cleanup reference appears in the handle cleanup reference queue, invoking a handle cleanup reference cleanup method;

the handle cleanup reference cleanup method decrementing a remote object handle reference count on a remote object handle associated with the handle cleanup reference;

the handle cleanup reference cleanup method removing the handle cleanup reference from a set of handle cleanup references monitored by the remote object handle manager;

when the remote object handle reference count after decrementing is zero, sending a digital message to the remote query processor to release an associated exported object handle; and

when the remote object handle reference count after decrementing is greater than zero, maintaining a strong reference to the remote object handle in order to ensure liveness for dependent client objects.

21. The nontransitory computer readable medium of claim 17, wherein the exported object handle maintains a strong reference to an update propagation graph created on the server computer.

22. The nontransitory computer readable medium of claim 17, wherein the relationship between a remote object handle and its associated exported object extends an update propagation graph across at least one of multiple remote query processors and multiple clients.

23. The nontransitory computer readable medium of claim 17, wherein the remote object handle invokes one or more methods on an exported object and delivers return values as copied objects or remote object handles of exported objects.

24. The nontransitory computer readable medium of claim 17, wherein the remote client computer and the server computer are different computers.

25. The system of claim 1, wherein the remote object handle manager monitoring of client object liveness comprises:

the remote object handle manager monitoring a handle cleanup reference queue;

after a handle cleanup reference appears in the handle cleanup reference queue, invoking a handle cleanup reference cleanup method;

the handle cleanup reference cleanup method removing the handle cleanup reference from a set of handle cleanup references monitored by the remote object handle manager, thereby eliminating all strong references to the handle cleanup reference; and

the handle cleanup reference cleanup method sending a digital message to the remote query processor to release an associated exported object handle.

26. The system of claim 1, wherein the remote object handle manager monitoring of client object liveness comprises:

the remote object handle manager monitoring a handle cleanup reference queue;

after a handle cleanup reference appears in the handle cleanup reference queue, invoking a handle cleanup reference cleanup method;

the handle cleanup reference cleanup method decrementing a remote object handle reference count on a remote object handle associated with the handle cleanup reference;

the handle cleanup reference cleanup method removing the handle cleanup reference from a set of handle cleanup references monitored by the remote object handle manager; and

when the remote object handle reference count after decrementing is zero, sending a digital message to the remote query processor to release an associated exported object handle.

27. The system of claim 1, wherein the remote query processor preserving a liveness of the exported object comprises:

maintaining a reference count associated with the exported object;

decrementing the reference count associated with the exported object after receipt of a release notification from the remote table handle manager;

decrementing the reference count associated with the exported object after the liveness indication system determines the remote client computer is not connected;

when the remote object handle reference count after decrementing is zero, removing a strong reference to the exported object from the exported object handle; and

when the remote object handle reference count after decrementing is greater than zero, maintaining a strong reference to the exported object from the exported object handle.

28. A method for monitoring liveness of parent objects and child objects participating in an update propagation graph on a query processing computer having parent listener objects and child listener objects, the method comprising:

maintaining continued liveness of said parent objects so long as their child objects have continued liveness;

permitting termination of liveness of dependent child objects when their parent objects have continued liveness;

maintaining continued liveness of said parent listener objects so long as their child objects have continued liveness; and

permitting termination of liveness of dependent child listener objects when their parent objects have continued liveness.

29. The method of claim 28 wherein:

said child objects are only weakly reachable from their parent objects; and

said parent objects are strongly reachable from their child objects.

30. The method of claim 9 wherein the preserving a liveness of the exported object comprises:

maintaining a reference count associated with the exported object;

decrementing the reference count associated with the exported object after receipt of a release notification from the remote table handle manager;

decrementing the reference count associated with the exported object after the liveness indication system determines the remote client computer is not connected;

when the remote object handle reference count after decrementing is zero, removing a strong reference to the exported object from the exported object handle; and

when the remote object handle reference count after decrementing is greater than zero, maintaining a strong reference to the exported object from the exported object handle.

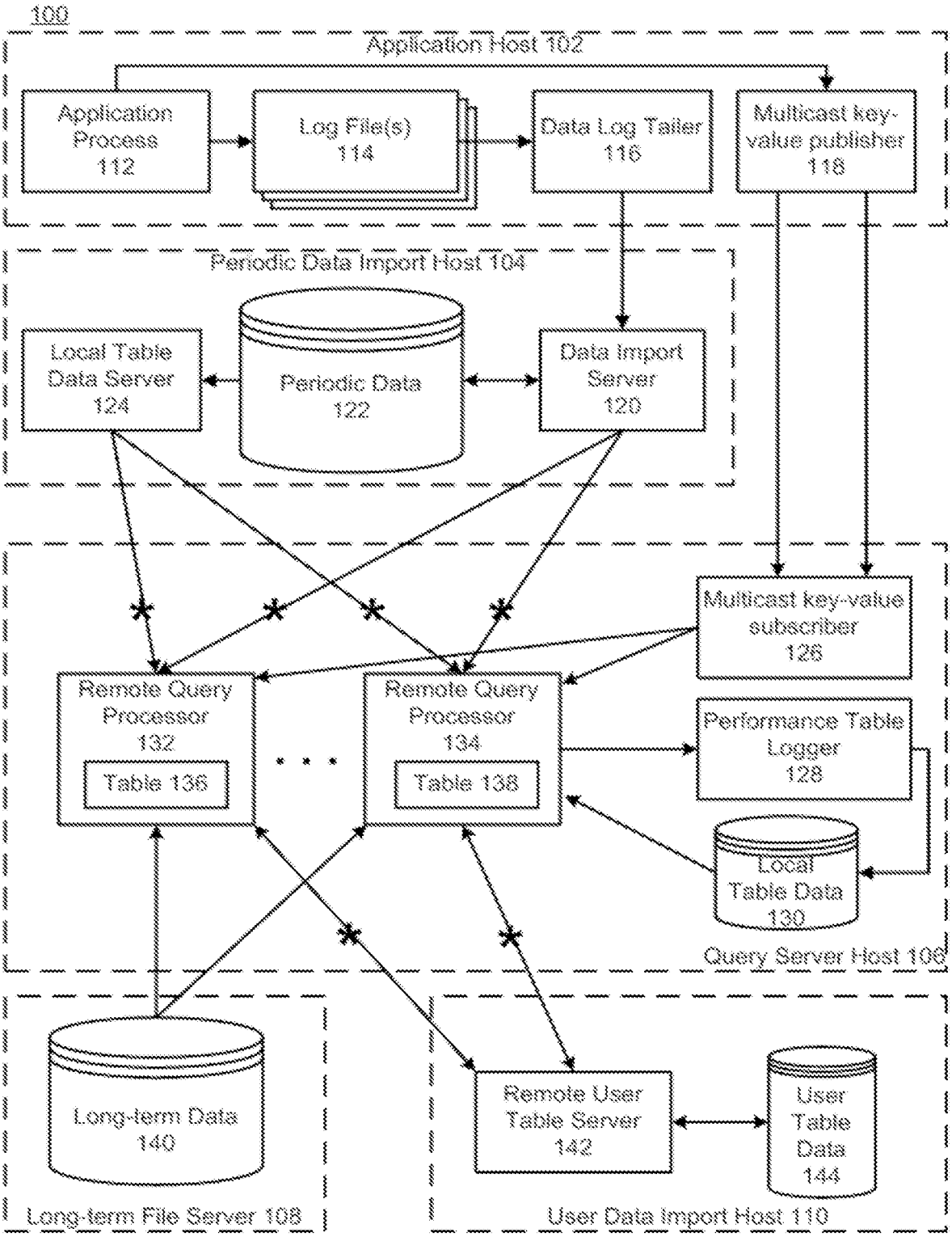
100
Application Host 102
Application Process 112
Log File(s) 114
Data Log Tailer 116
Multicast key-value publisher 118
Periodic Data Import Host 104
Local Table Data Server 124
Periodic Data 122
Data Import Server 120
Multicast key-value subscriber 126
Remote Query Processor 132
Table 136
Remote Query Processor 134
Table 138
Performance Table Logger 128
Local Table Data 130
Query Server Host 106
Long-term Data 140
Long-term File Server 108
Remote User Table Server 142
User Table Data 144
User Data Import Host 110

FIG. 1

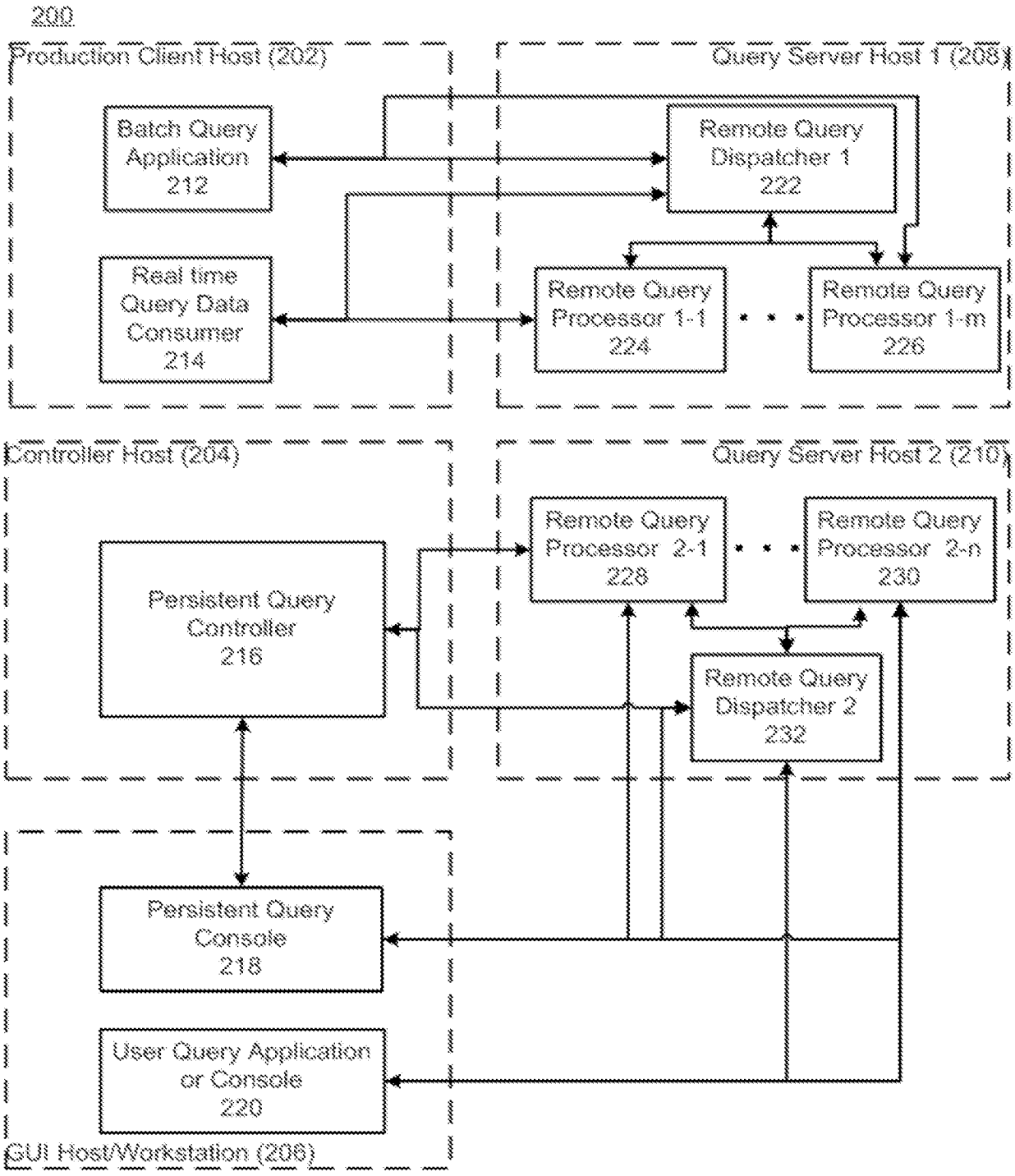
200
Production Client Host (202)
Batch Query Application 212
Real time Query Data Consumer 214
Query Server Host 1 (208)
Remote Query Dispatcher 1 222
Remote Query Processor 1-1 224
Remote Query Processor 1-m 226
Controller Host (204)
Persistent Query Controller 216
Query Server Host 2 (210)
Remote Query Processor 2-1 228
Remote Query Processor 2-n 230
Remote Query Dispatcher 2 232
Persistent Query Console 218
User Query Application or Console 220
GUI Host/Workstation (206)

FIG. 2

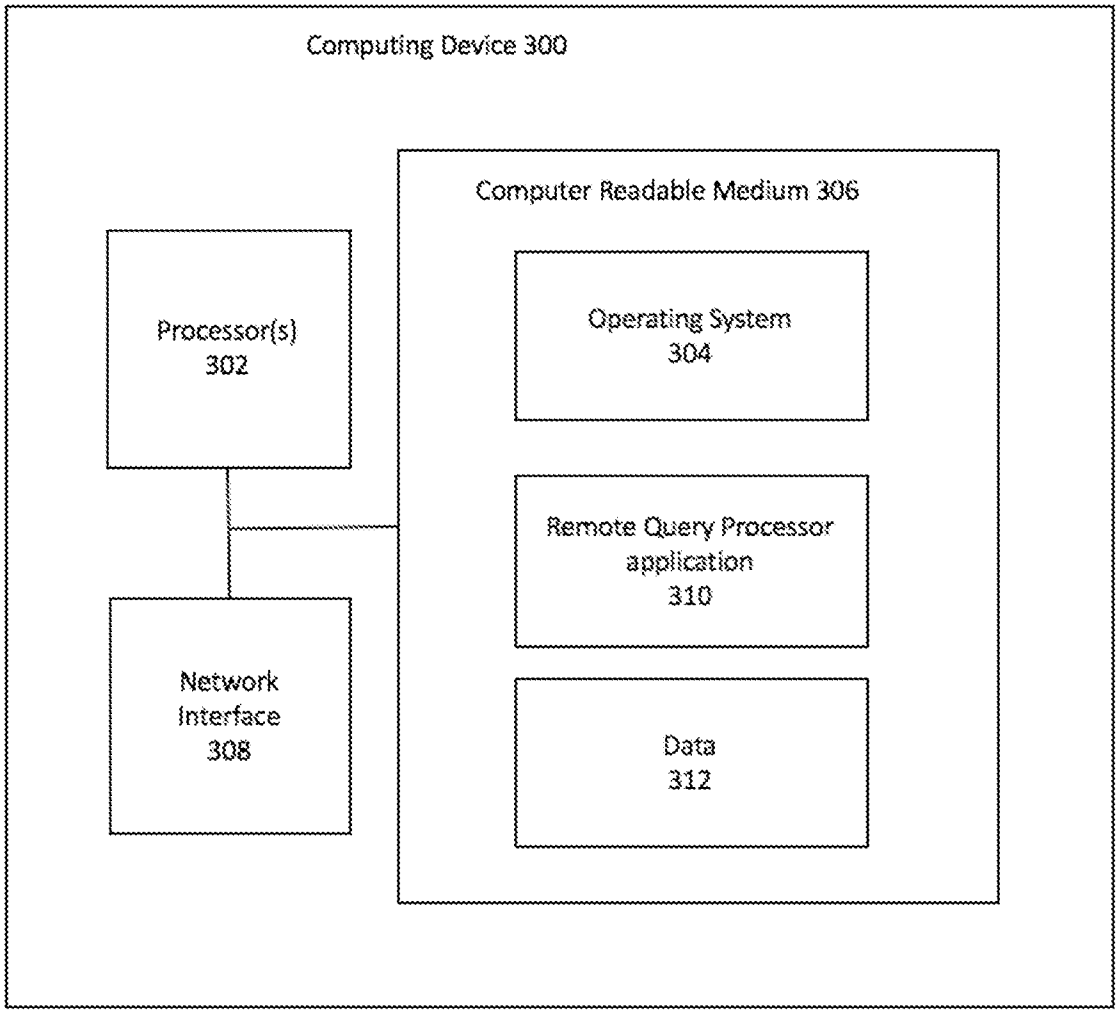
Computing Device 300
Computer Readable Medium 306
Processor(s)
302
Operating System
304
Remote Query Processor
application
310
Network
Interface
308
Data
312

FIG. 3

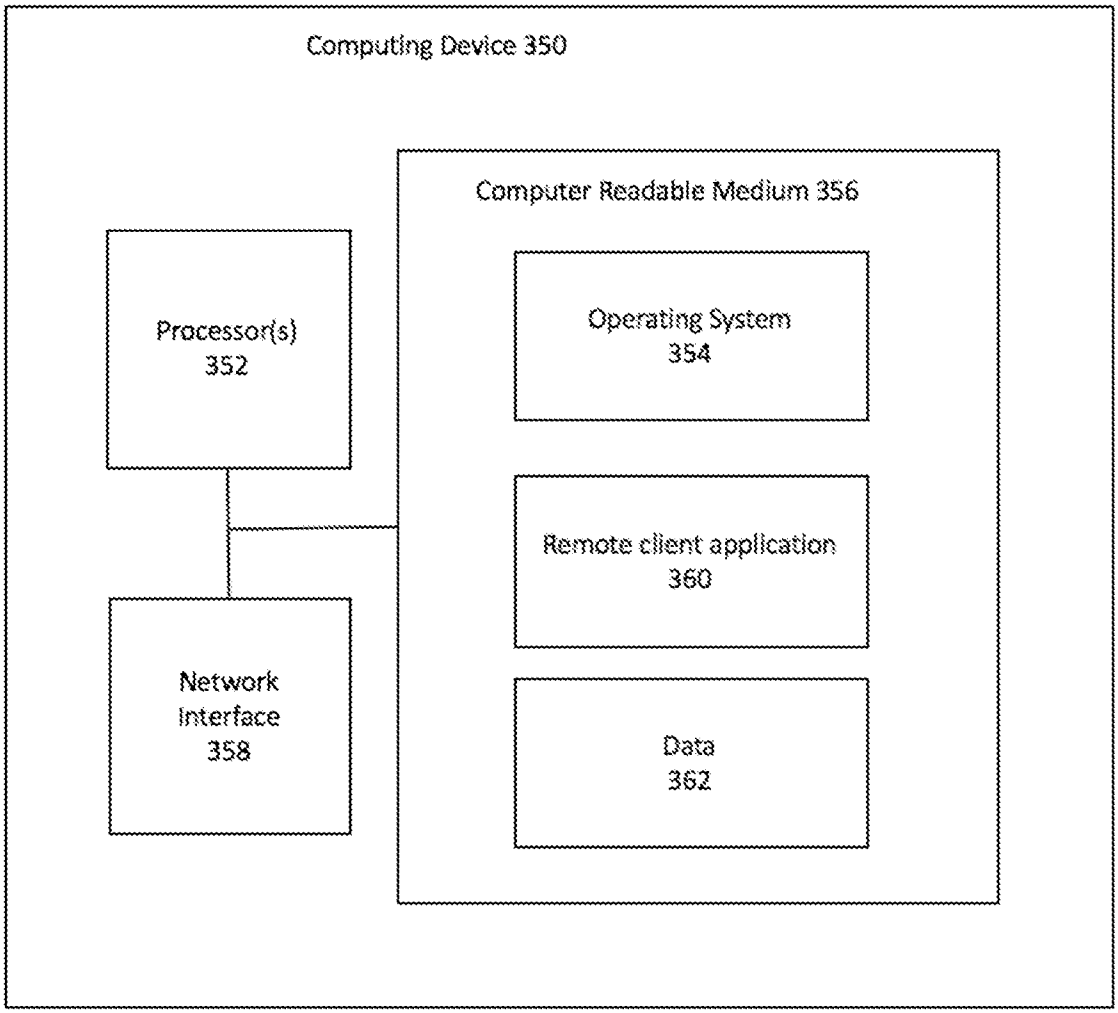
Computing Device 350
Computer Readable Medium 356
Processor(s)
352
Operating System
354
Remote client application
360
Network
Interface
358
Data
362

FIG. 3A

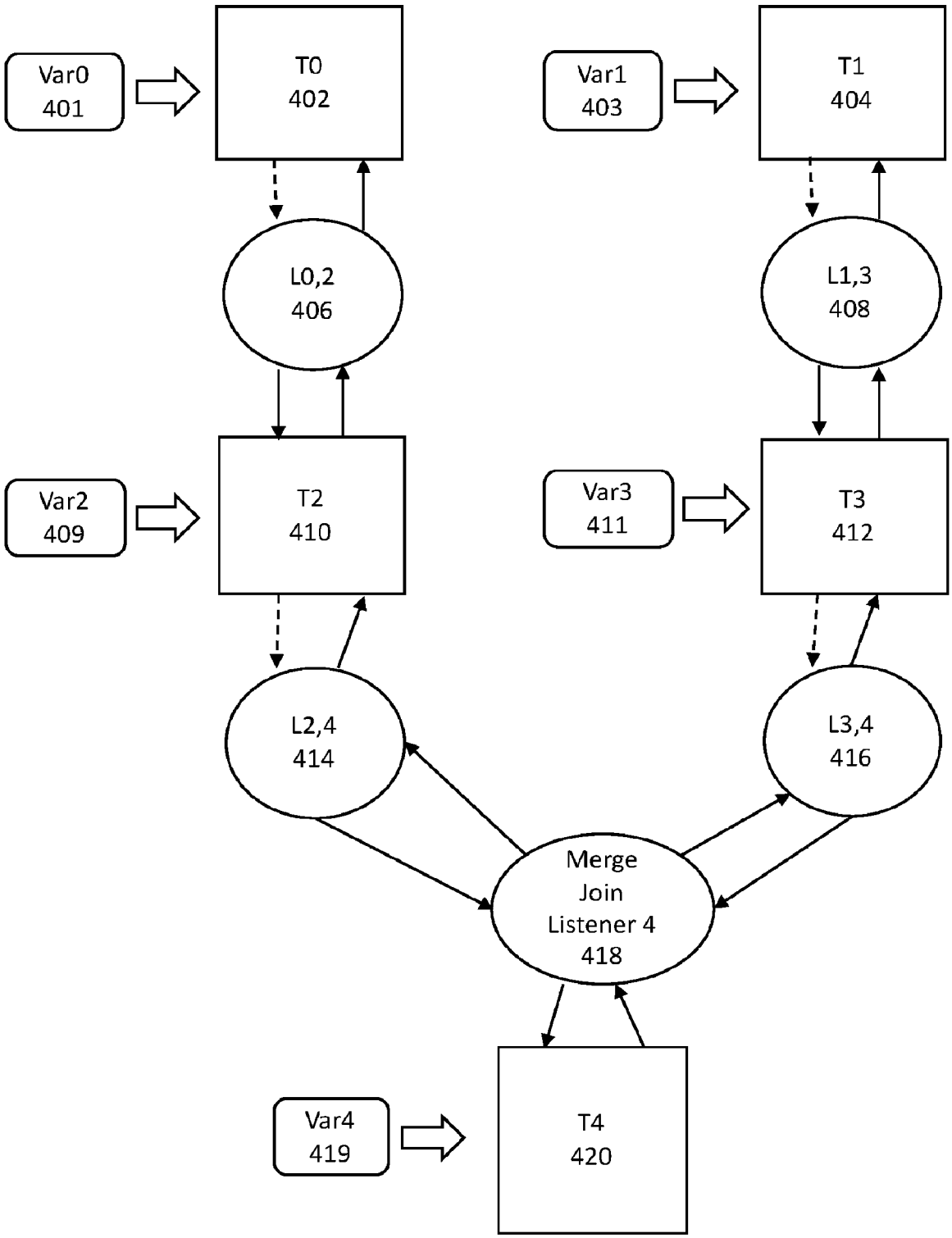
Var0
401
T0
402
Var1
403
T1
404
L0,2
406
L1,3
408
Var2
409
T2
410
Var3
411
T3
412
L2,4
414
L3,4
416
Merge
Join
Listener 4
418
Var4
419
T4
420

FIG. 4

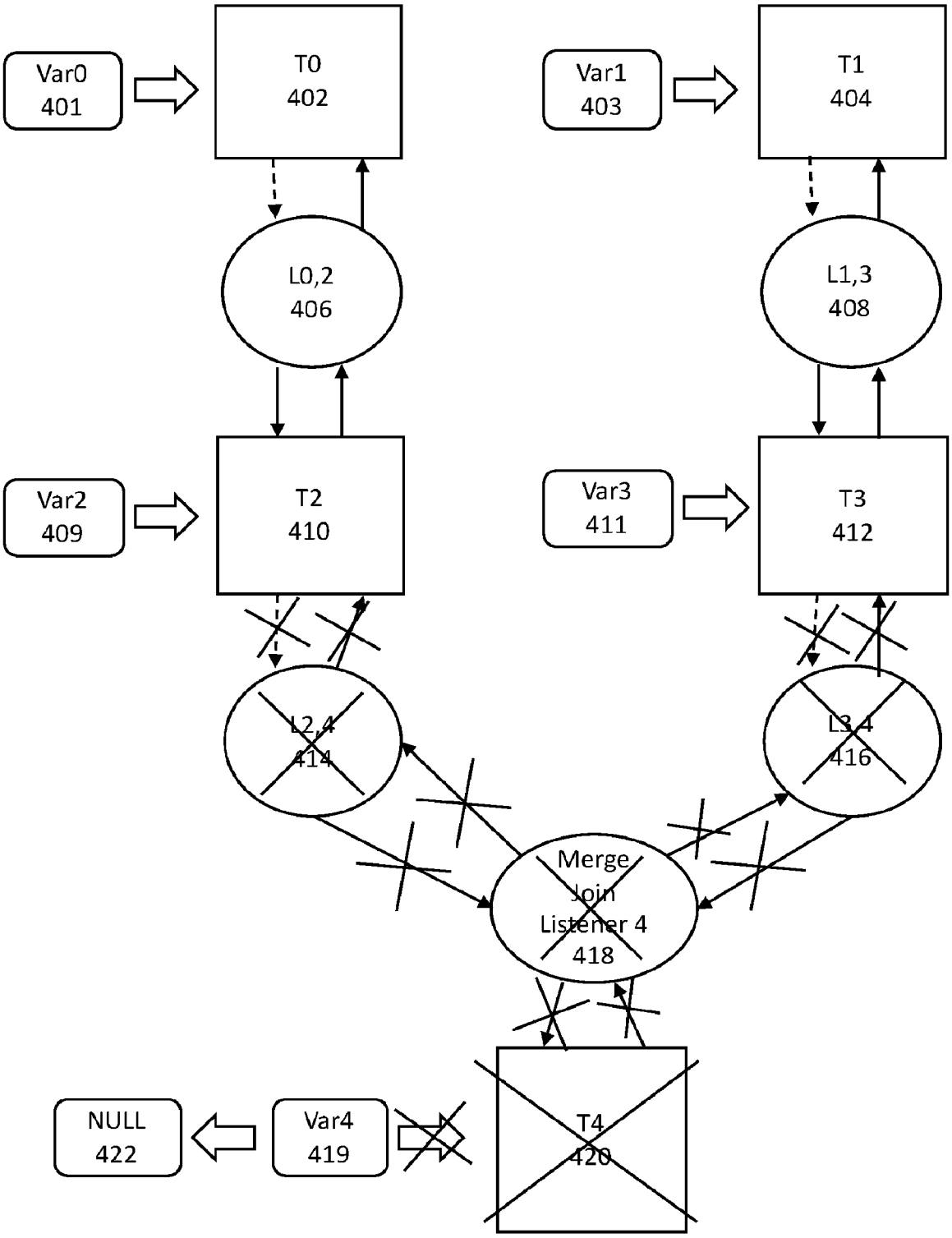
Var0
401
T0
402
Var1
403
T1
404
L0,2
406
L1,3
408
Var2
409
T2
410
Var3
411
T3
412
L2,4
414
L3,4
416
Merge
Join
Listener 4
418
NULL
422
Var4
419
T4
420

FIG. 4A

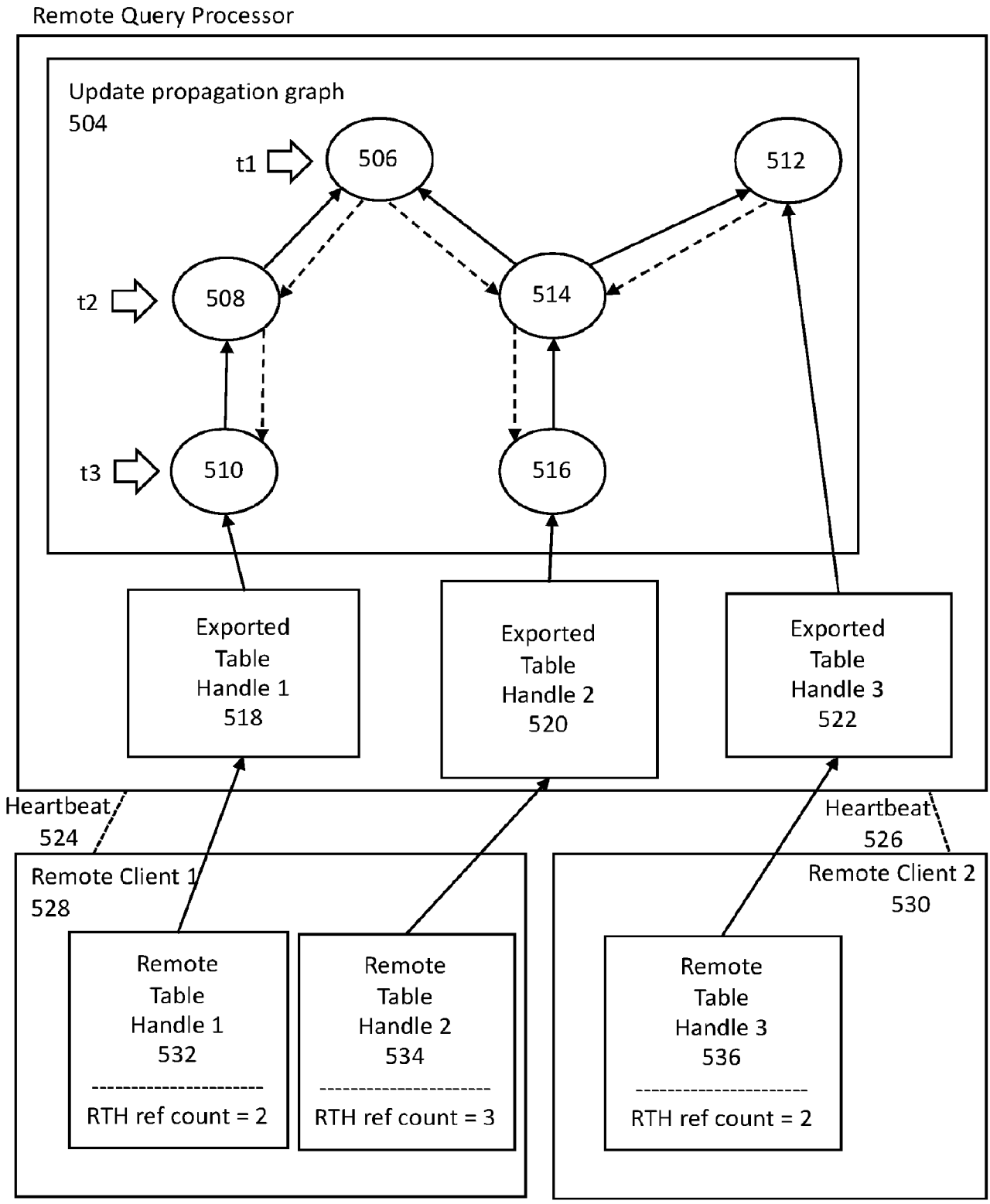
Remote Query Processor
Update propagation graph
504
t1
506
512
t2
508
514
t3
510
516
Exported
Table
Handle 1
518
Exported
Table
Handle 2
520
Exported
Table
Handle 3
522
Heartbeat
524
Heartbeat
526
Remote Client 1
528
Remote Client 2
530
Remote
Table
Handle 1
532
RTH ref count = 2
Remote
Table
Handle 2
534
RTH ref count = 3
Remote
Table
Handle 3
536
RTH ref count = 2

FIG. 5

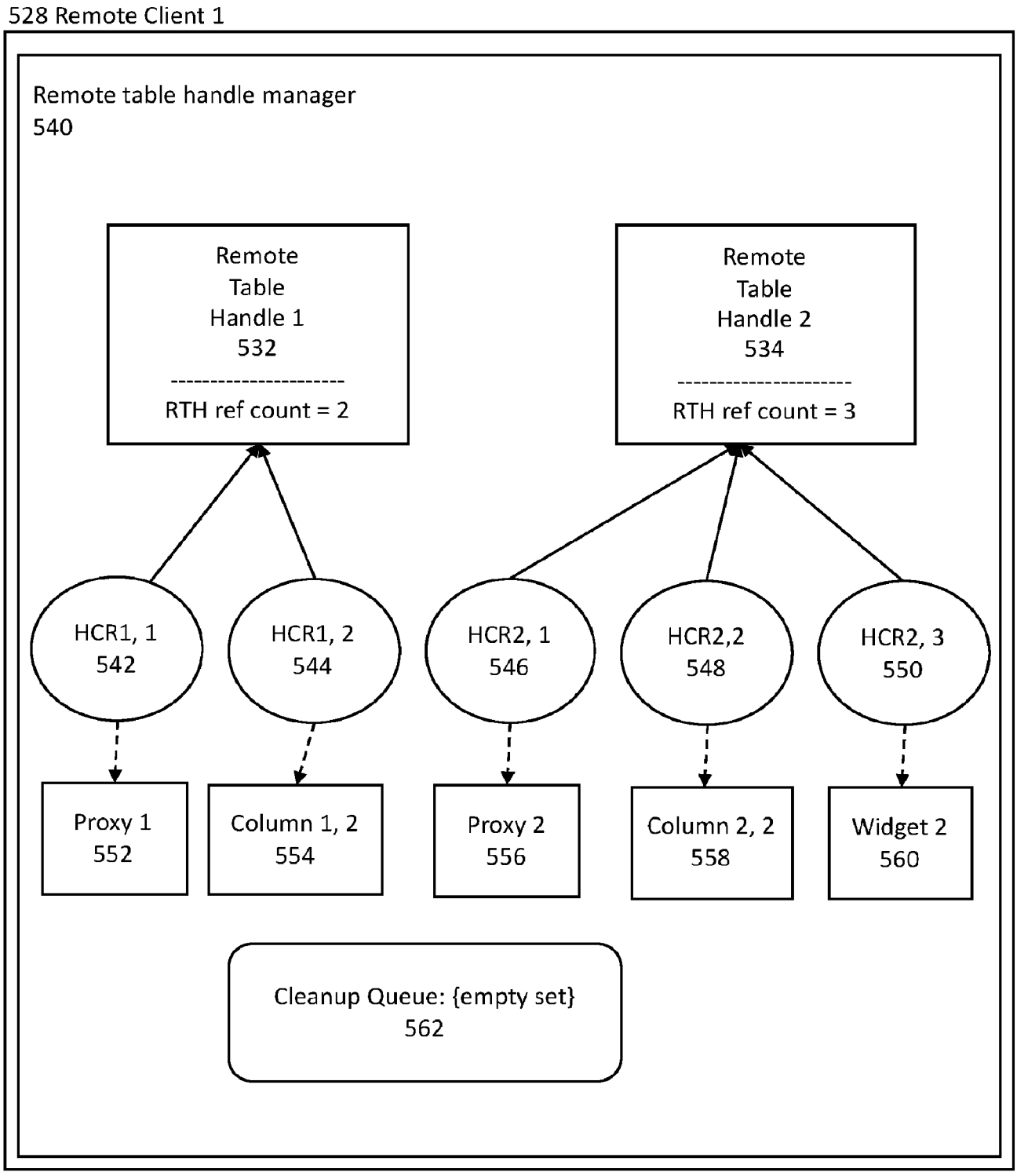
528 Remote Client 1
Remote table handle manager
540
Remote
Table
Handle 1
532
RTH ref count = 2
Remote
Table
Handle 2
534
RTH ref count = 3
HCR1, 1
542
HCR1, 2
544
HCR2, 1
546
HCR2,2
548
HCR2, 3
550
Proxy 1
552
Column 1, 2
554
Proxy 2
556
Column 2, 2
558
Widget 2
560
Cleanup Queue: {empty set}
562

FIG. 5A

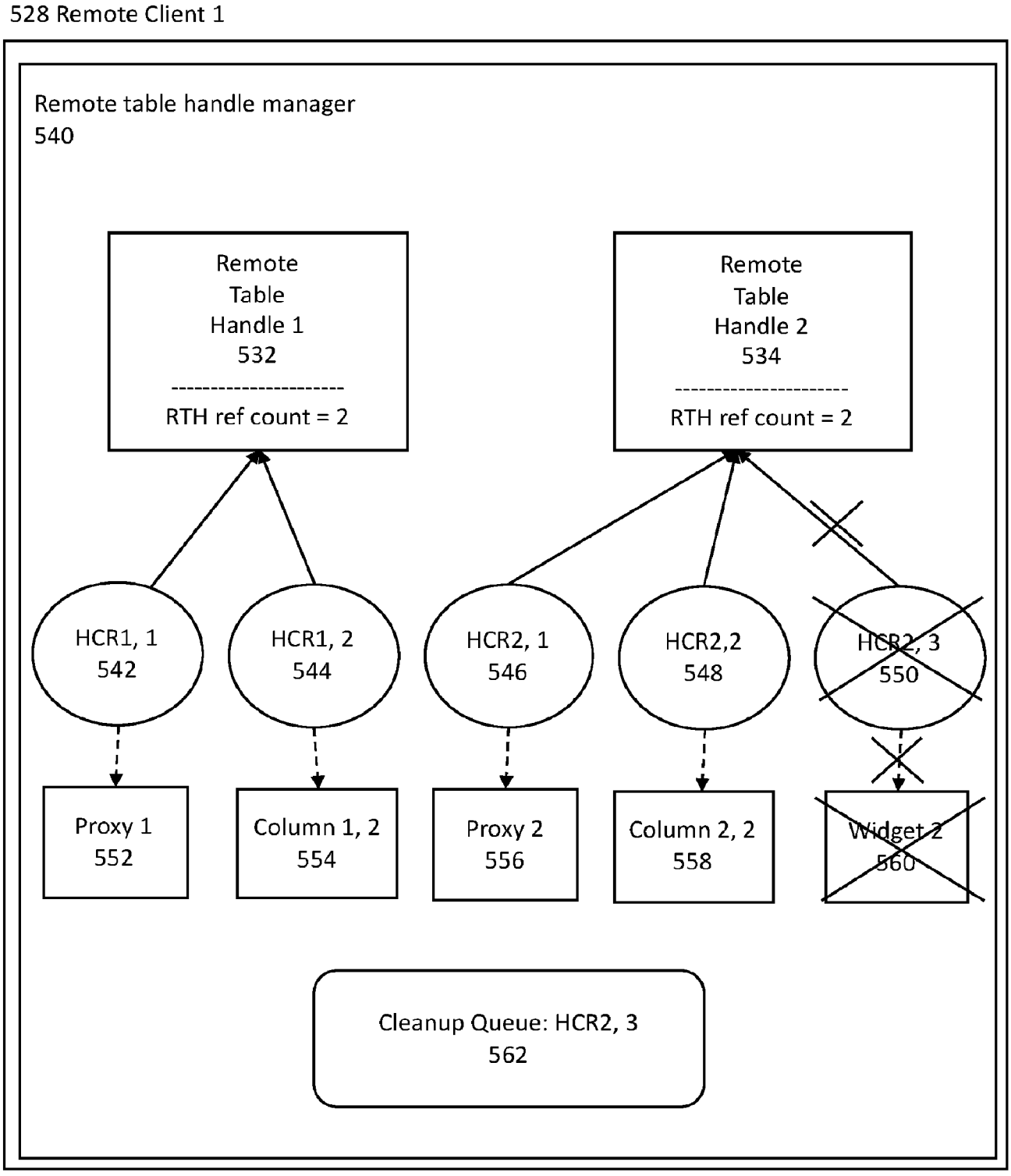
528 Remote Client 1
Remote table handle manager
540
Remote
Table
Handle 1
532
RTH ref count = 2
Remote
Table
Handle 2
534
RTH ref count = 2
HCR1, 1
542
HCR1, 2
544
HCR2, 1
546
HCR2,2
548
HCR2, 3
550
Proxy 1
552
Column 1, 2
554
Proxy 2
556
Column 2, 2
558
Widget 2
560
Cleanup Queue: HCR2, 3
562

FIG. 5B

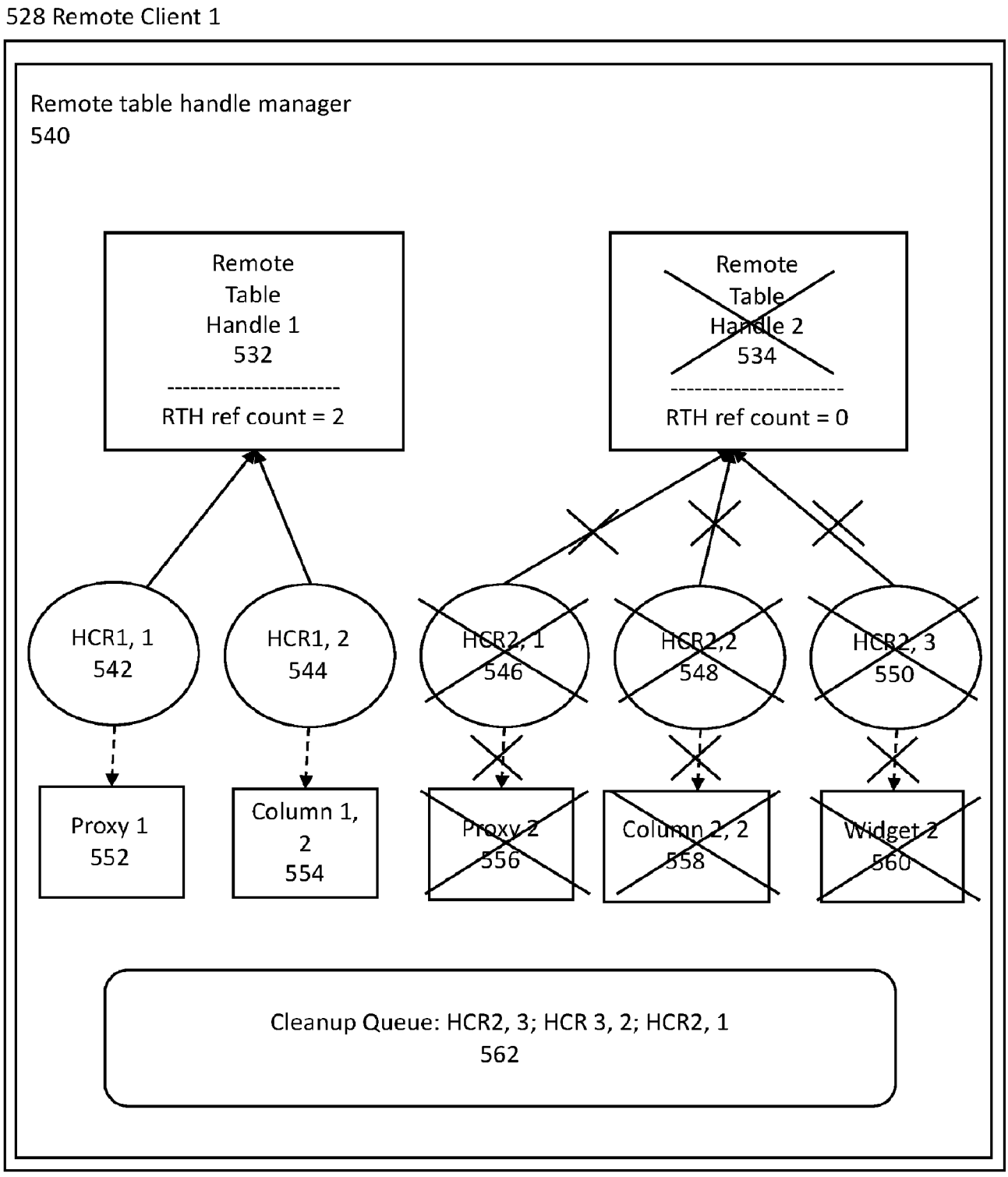
528 Remote Client 1
Remote table handle manager
540
Remote
Table
Handle 1
532
RTH ref count = 2
Remote
Table
Handle 2
534
RTH ref count = 0
HCR1, 1
542
HCR1, 2
544
HCR2, 1
546
HCR2,2
548
HCR2, 3
550
Proxy 1
552
Column 1,
2
554
Proxy 2
556
Column 2, 2
558
Widget 2
560
Cleanup Queue: HCR2, 3; HCR 3, 2; HCR2, 1
562

FIG. 5C

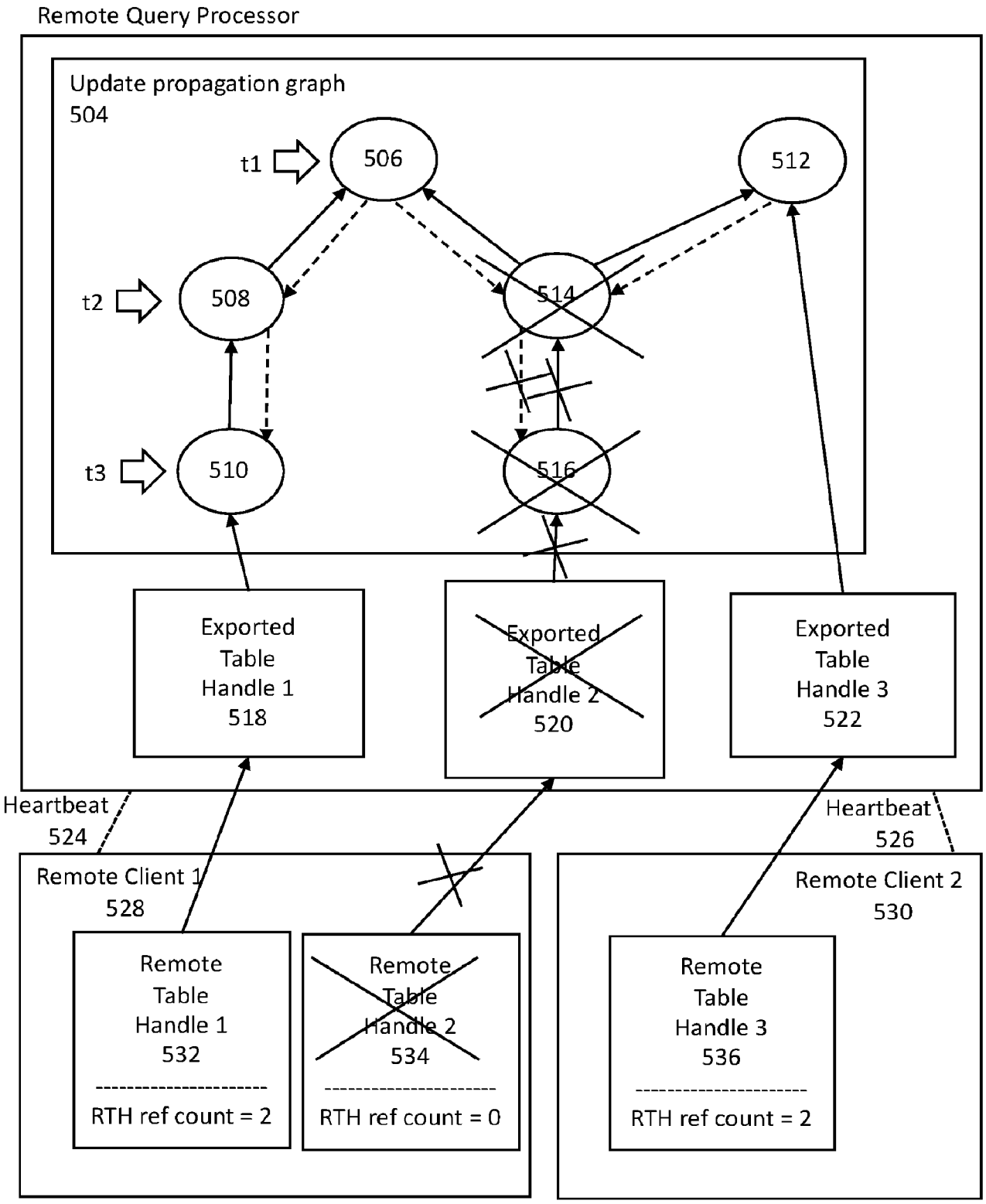
Remote Query Processor
Update propagation graph
504
t1
506
512
t2
508
514
t3
510
516
Exported
Table
Handle 1
518
Exported
Table
Handle 2
520
Exported
Table
Handle 3
522
Heartbeat
524
Heartbeat
526
Remote Client 1
528
Remote Client 2
530
Remote
Table
Handle 1
532
RTH ref count = 2
Remote
Table
Handle 2
534
RTH ref count = 0
Remote
Table
Handle 3
536
RTH ref count = 2

FIG. 5D

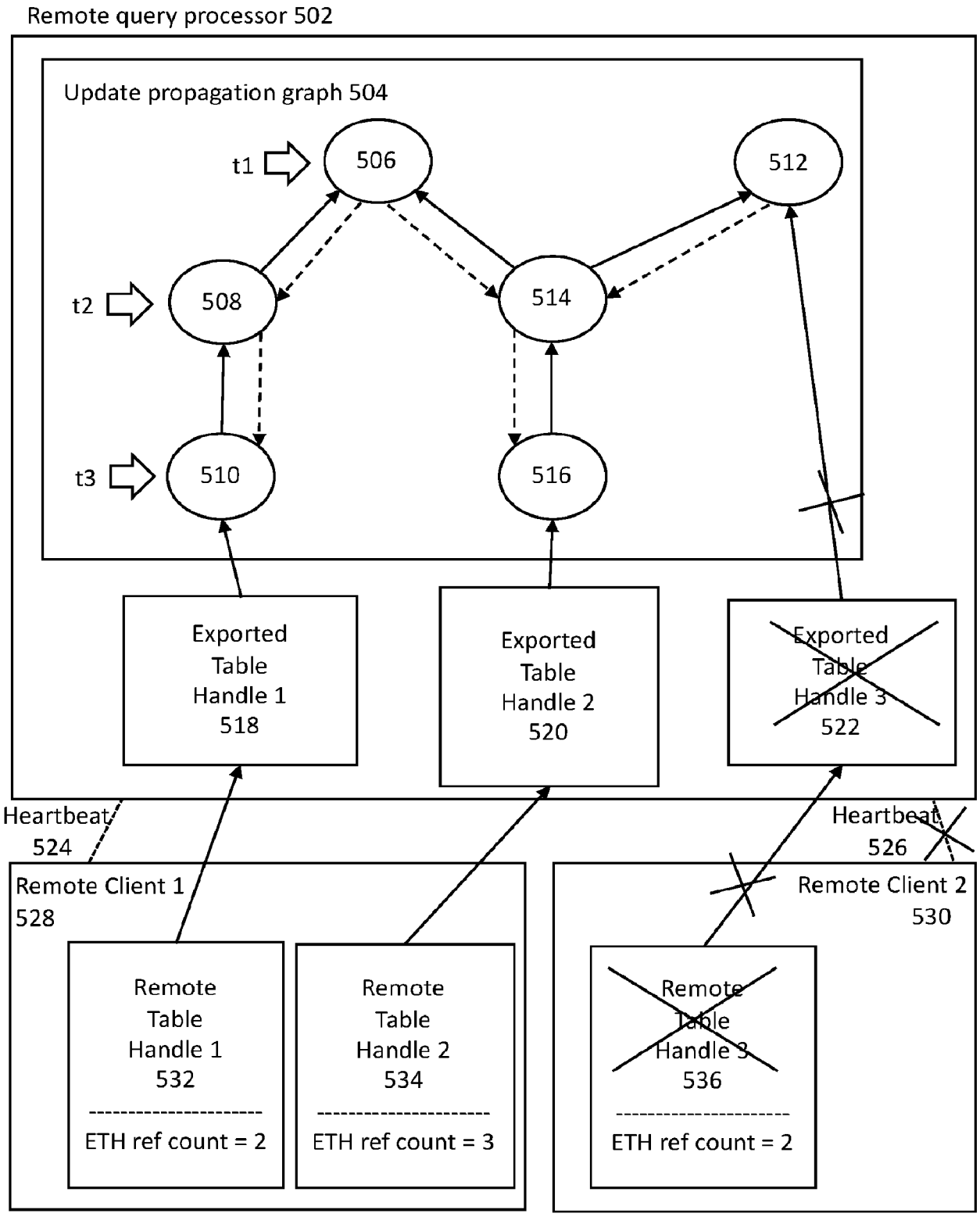
Remote query processor 502
Update propagation graph 504
t1
506
512
t2
508
514
t3
510
516
Exported Table Handle 1 518
Exported Table Handle 2 520
Exported Table Handle 3 522
Heartbeat 524
Heartbeat 526
Remote Client 1 528
Remote Client 2 530
Remote Table Handle 1 532
ETH ref count = 2
Remote Table Handle 2 534
ETH ref count = 3
Remote Table Handle 3 536
ETH ref count = 2

FIG. 5E

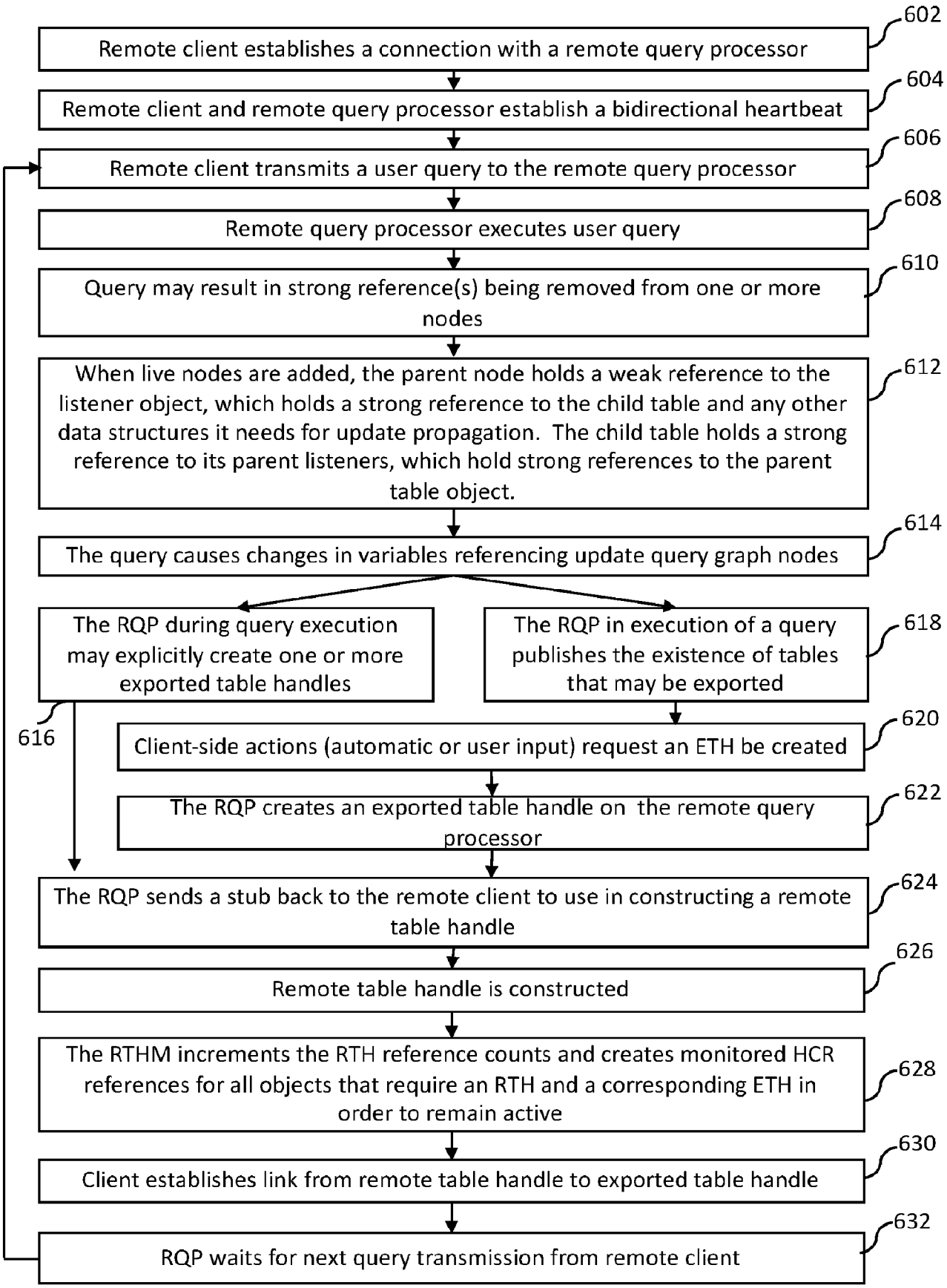
600 Remote Table Handle Establishment
Remote client establishes a connection with a remote query processor
602
Remote client and remote query processor establish a bidirectional heartbeat
604
Remote client transmits a user query to the remote query processor
606
Remote query processor executes user query
608
Query may result in strong reference(s) being removed from one or more nodes
610
When live nodes are added, the parent node holds a weak reference to the listener object, which holds a strong reference to the child table and any other data structures it needs for update propagation. The child table holds a strong reference to its parent listeners, which hold strong references to the parent table object.
612
The query causes changes in variables referencing update query graph nodes
614
The RQP during query execution may explicitly create one or more exported table handles
The RQP in execution of a query publishes the existence of tables that may be exported
618
616
Client-side actions (automatic or user input) request an ETH be created
620
The RQP creates an exported table handle on the remote query processor
622
The RQP sends a stub back to the remote client to use in constructing a remote table handle
624
Remote table handle is constructed
626
The RTHM increments the RTH reference counts and creates monitored HCR references for all objects that require an RTH and a corresponding ETH in order to remain active
628
Client establishes link from remote table handle to exported table handle
630
RQP waits for next query transmission from remote client
632

FIG. 6

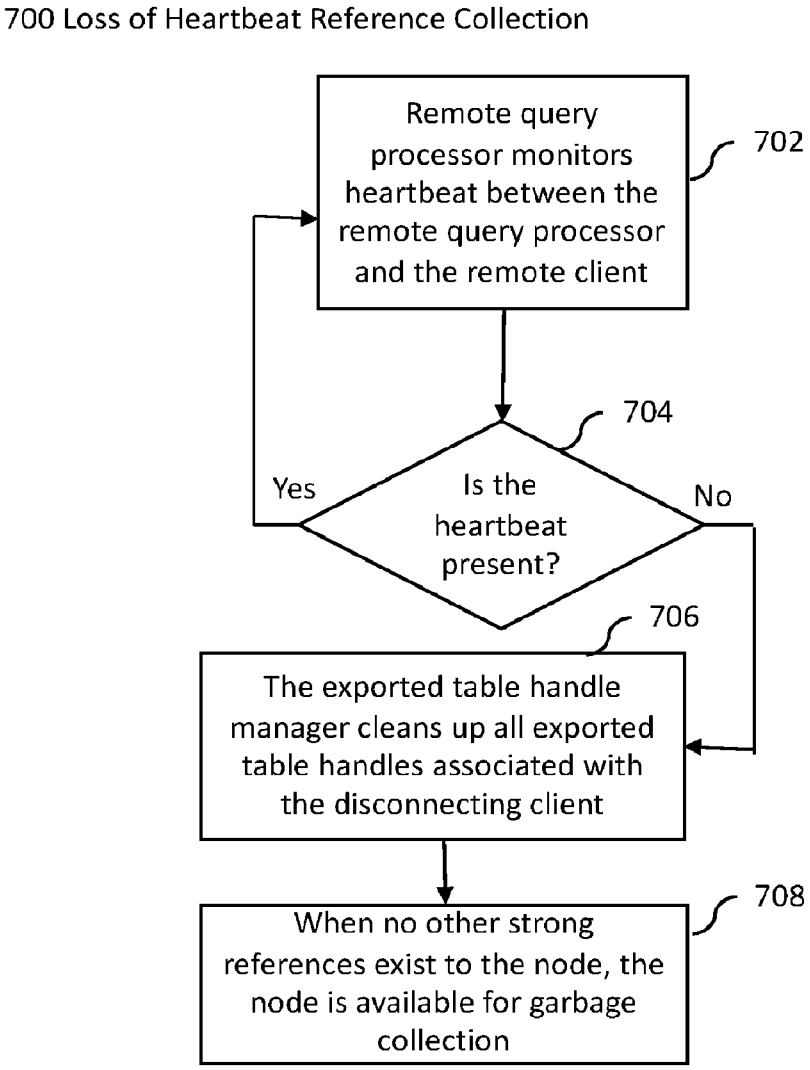
700 Loss of Heartbeat Reference Collection
Remote query processor monitors heartbeat between the remote query processor and the remote client
702
704
Is the heartbeat present?
Yes
No
706
The exported table handle manager cleans up all exported table handles associated with the disconnecting client
When no other strong references exist to the node, the node is available for garbage collection
708

FIG. 7

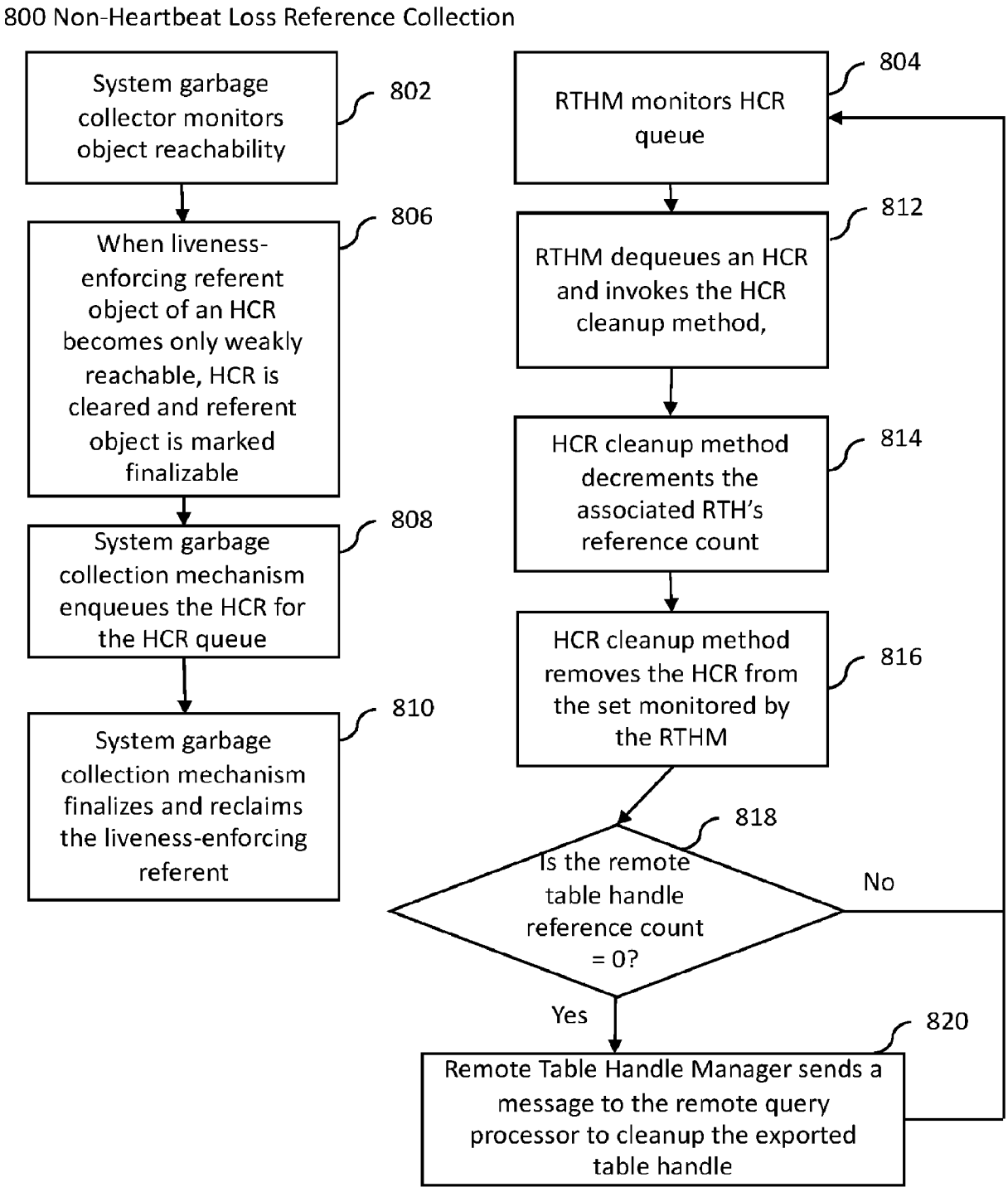
800 Non-Heartbeat Loss Reference Collection
System garbage collector monitors object reachability
802
When liveness-enforcing referent object of an HCR becomes only weakly reachable, HCR is cleared and referent object is marked finalizable
806
System garbage collection mechanism enqueues the HCR for the HCR queue
808
System garbage collection mechanism finalizes and reclaims the liveness-enforcing referent
810
RTHM monitors HCR queue
804
RTHM dequeues an HCR and invokes the HCR cleanup method,
812
HCR cleanup method decrements the associated RTH's reference count
814
HCR cleanup method removes the HCR from the set monitored by the RTHM
816
Is the remote table handle reference count = 0?
818
No
Yes
Remote Table Handle Manager sends a message to the remote query processor to cleanup the exported table handle
820

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 2016/032587

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/02 *(2006.01)*
G06F 17/30 *(2006.01)*

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 12/00-12/02, 17/00-17/30, H04L 9/00-9/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PatSearch (RUPTO internal), USPTO, PAJ, K-PION, Esp@cenet, Information Retrieval System of FIPS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5960087 A (SUN MICROSYSTEMS, INC.) 28.09.1999, abstract, col. 4, lines 35-49, col. 6, line 66-col. 7, line 25, col. 13, lines 62-67, col. 15, line 38-col. 16, lines 3, 9-32, col. 17, line 58-col. 18, line 4, col. 22, line 26-col. 23, line 13	1-27, 30
Y	US 2002/0002576 A1 (SUN MICROSYSTEMS, INC.) 03.01.2002, [0014], [0039]	1-27, 30
X	US 8838656 B1 (HISCAMP SYSTEMS, INC.) 16.09.2014, abstract, claims 1, 5, 15, 20	28, 29

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 July 2016 (20.07.2016)	28 July 2016 (28.07.2016)

Name and mailing address of the ISA/RU:	Authorized officer
Federal Institute of Industrial Property, Berezhkovskaya nab., 30-1, Moscow, G-59, GSP-3, Russia, 125993 Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37	A. Tokarev Telephone No. 499-240-25-91