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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 December 2014 (31.12.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/209848 A1

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

(21) International Application Number:
PCT/US2014/043599

(22) International Filing Date:
23 June 2014 (23.06.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/841,045 28 June 2013 (28.06.2013) US
14/226,557 26 March 2014 (26.03.2014) US

(71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, M/S 5OP7, Redwood Shores, California 94065-1677 (US).

(72) Inventors: **HARDY, Alexandre**; 21 York Close, Howard Hamlet, University Drive, 7405 Pinelands (ZA). **TILAK, Omkar**; 1883 Hillebrant Place, Santa Clara, California 95050 (US).

(74) Agents: **NICHOLES, Christian A.** et al.; KILPATRICK TOWNSEND & STOCKTON LLP, Two Embarcadero Center, Eighth Floor, San Francisco, California 94111 (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, LT, 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 next page]

(54) Title: NAIVE, CLIENT-SIDE SHARDING WITH ONLINE ADDITION OF SHARDS

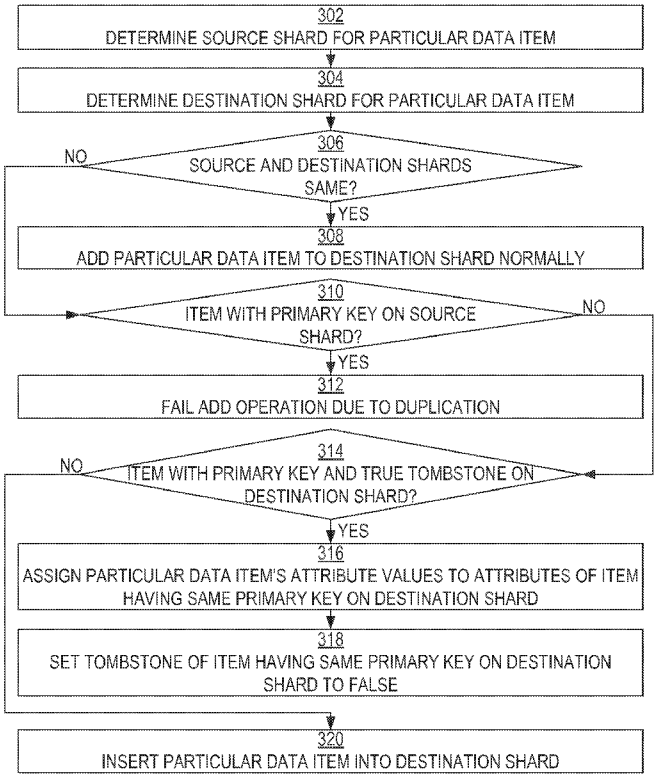

FIG. 3

(57) Abstract: Multiple clients can be enabled to perform operations relative to data items in a shard system asynchronously to each other without the use by those clients of exclusive locks. A rebalancing event, in which data items are redistributed automatically among a set of shards due to a modification of the quantity of shards in the system, can be performed without the use of exclusive locks by clients. Clients can continue to perform operations relative to at least some of the data items in the shard system even while rebalancing processes are redistributing at least some of the data items asynchronously during a system-wide rebalancing event. All of these benefits can be obtained without sacrificing data consistency within the shard system.

WO 2014/209848 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, 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))*

NAÏVE, CLIENT-SIDE SHARDING WITH ONLINE ADDITION OF SHARDS

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

CROSS-REFERENCES TO RELATED APPLICATIONS; PRIORITY CLAIM

[0001] The present application claims priority under 35 U.S.C. § 119 to U.S. Provisional Patent Application Serial No. 61/841,045, titled NAÏVE, CLIENT-SIDE SHARDING WITH ONLINE ADDITION OF SHARDS, filed on June 28, 2013, the entire contents of which are incorporated by reference herein; and U.S. Patent Application Serial No. 14/226,557, titled NAÏVE, CLIENT SIDE SHARDING WITH ONLINE ADDITION OF SHARDS, filed on March 26, 2014, the entire contents of which are incorporated by reference herein.

BACKGROUND

[0002] In a system of database shards, data items, such as records or rows or documents, can be distributed among multiple separate, independent databases, called shards. In such a system, data items typically are not, and are not permitted to be, duplicated among the shards. Thus, in such a system, a particular data item will be located on only one of the several shards at any given time. In order for a client to determine which of the several shards contains the particular data item, the client can input the data item's primary key—which uniquely identifies the particular data item—into a hash function. The hash function computes, based on the primary key, the identity of the shard on which the particular data item is currently stored. For example, a hash function might divide a numeric primary key by the quantity of shards in the system and then take the remainder (essentially a modulo operation) to be the identifier for the shard that

contains the particular data item. Using such a hash function to determine, in the first place, the shard on which each data item will be stored typically causes data items to be distributed relatively evenly among the shards.

[0003] Once the client has identified the shard upon which a particular data item is located, the client can perform operations, such as read, delete, or update operations, relative to the data item. Usually, a shard system will serve numerous clients concurrently, and these clients may each perform, asynchronously to each other, operations relative to separate data items. Potentially, multiple clients could inadvertently attempt to perform operations relative to the same data item simultaneously. If this scenario were permitted to occur unhindered, then the data item could become corrupted, making the state of the shard system inconsistent. Under one approach, in order to guarantee that multiple clients will not concurrently perform operations relative to the same data item, a client that seeks to perform an operation relative to a particular data item can first be required to acquire an exclusive lock on that particular data item. Each data item can be associated with a separate lock. A client is prevented from acquiring the exclusive lock on the particular data item if another client already holds that exclusive lock; under such circumstances, the client seeking to obtain the exclusive lock must wait for the lock-holding client to release the exclusive lock. While a client is holding the exclusive lock on a particular data item, that client alone can perform operations relative to the particular data item. When the client has finished performing operations relative to the particular data item, the client can release the exclusive lock on the particular data item, thereby making the particular data item available for access by other clients.

[0004] As the quantity of data stored within the shard system grows, the capacity of the existing shards in the system might become inadequate to contain all of the data that is going to be stored in the system. It can be desirable, under those circumstances, to add one or more new shards to the system. The addition of new shards can involve the addition of new hardware computing and storage devices to contain and manage new databases. In order to attempt to balance the client access load among the shards, so that no one subset of shards becomes disproportionately burdened with client requests, the addition of new shards to the system can precipitate a redistribution of the system's stored data items among the augmented group of shards. The redistribution event, or rebalancing event, can cause data items that were formerly

stored on one shard to be re-located to another shard—potentially, but not necessarily, to a newly added shard. Under one approach, rebalancing processes can obtain exclusive locks on the data items that are to be moved. After obtaining the exclusive locks on the data items, the rebalancing processes can move those data items from old shards to the new shards that have been determined by a revised hash function to be the destination for those data items. After moving the data items, the rebalancing processes can release the exclusive locks on those data items.

[0005] For as long as exclusive locks have existed, some drawbacks have attended their uses. One drawback is that while a data item's exclusive lock is held by a process, no other process can access that data item. Thus, under the lock-using rebalancing approach discussed above, clients may be largely unable to perform operations relative to the shard system while the rebalancing event proceeds; no client can obtain an exclusive lock on a data item while a rebalancing process holds that exclusive lock. Even when the rebalancing event is not ongoing, the overhead involved in clients' acquisition, maintenance, and release of locks—to guard against concurrent multiple client access—can be significant and burdensome. Even ignoring the effects of rebalancing events, the use of locks within a shard system can negatively impact system efficiency and performance. Perhaps worse still, unexpected failures within the shard system can cause a lock-holding process (which could be, for example, a client or a rebalancing process) to freeze up or otherwise quit functioning properly. Under such circumstances, the non-functional process may retain an exclusive lock on a particular data item until some timer expires, at which time the non-functional process may be terminated, and the locks it held forcibly released. Other processes, including clients and rebalancing processes, are consequently forced to wait for the timer's expiration before proceeding with their intended tasks relative to that particular data item. Especially if the operations to be performed relative to the particular data item are just one step within a strictly ordered sequence of operations to be performed relative to multiple separate data items, such forced waiting can cause the performance of the entire shard system to degrade noticeably. Dependencies imposed by orders in which operations often need to be performed can cause these kinds of complications to cascade.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] FIG. 1 is a block diagram illustrating an example of a scalable shard system in which

multiple clients can access data items that have been distributed among multiple database shards, according to an embodiment of the invention;

[0007] FIG. 2 is a state diagram that illustrates the various states in which system can exist at various moments in time, and the possible transitions between those states, according to an embodiment of the invention;

[0008] FIG. 3 is a flow diagram that illustrates an example of a technique for performing an add operation while in the rebalancing state, according to an embodiment of the invention;

[0009] FIG. 4 is a flow diagram that illustrates an example of a technique for performing an update operation while in the rebalancing state, according to an embodiment of the invention;

[0010] FIG. 5 is a flow diagram that illustrates an example of a technique for performing a delete operation while in the rebalancing state, according to an embodiment of the invention;

[0011] FIGs. 6A-6B are flow diagrams that illustrate an example of a technique for performing a get operation while in the rebalancing state, according to an embodiment of the invention;

[0012] FIG. 7 is a flow diagram that illustrates an example of a technique for performing a rebalancing operation while in the rebalancing state (initially), according to an embodiment of the invention;

[0013] FIG. 8 is a flow diagram that illustrates an example of a technique for performing a query operation while in the rebalancing state, according to an embodiment of the invention;

[0014] FIG. 9 depicts a simplified diagram of a distributed system for implementing one of the embodiments.

[0015] FIG. 10 is a simplified block diagram of components of a system environment by which services provided by the components of an embodiment system may be offered as cloud services, in accordance with an embodiment of the present disclosure.

[0016] FIG. 11 illustrates an example of a computer system in which various embodiments of the present invention may be implemented.

DETAILED DESCRIPTION

[0017] In the following description, for the purposes of explanation, specific details are set

forth in order to provide a thorough understanding of embodiments of the invention. However, it will be apparent that the invention may be practiced without these specific details.

[0018] According to an embodiment of the invention, multiple clients can be enabled to perform operations relative to data items in a shard system asynchronously to each other without the use by those clients of exclusive locks. Furthermore, according to an embodiment of the invention, a rebalancing event, in which data items are redistributed automatically among a set of shards due to a modification (addition or impending removal) of the quantity of shards in the system, can be performed without the use of exclusive locks by clients. Additionally, in an embodiment of the invention, clients can continue to perform operations relative to at least some of the data items in the shard system even while rebalancing processes are redistributing at least some of the data items asynchronously during a system-wide rebalancing event. The programmatic code that provides these features can be located exclusively on the clients rather than the shard servers. All of these benefits can be obtained without sacrificing data consistency within the shard system.

EXAMPLE SCALABLE SHARD SYSTEM

[0019] FIG. 1 is a block diagram illustrating an example of a scalable shard system 100 in which multiple clients can access data items that have been distributed among multiple database shards, according to an embodiment of the invention. Shard system 100 includes clients 102A-N and shards 104A-N. The quantities of clients and shards in system 100 can vary. Each of shards 104A-N can be a separate and independent database that does not need to be aware of any other shard within system 100. Each of shards 104A-N can include a separate database server and relational database, for example. Although databases are discussed herein as a concrete example, embodiments of the invention can be applied to a variety of kinds of data repositories (e.g., Lightweight Directory Access Protocol (LDAP) directories, flat files, associative memories, etc.) other than databases. Each of clients 102A-N can be a separate computing system that can operate independently of each other of clients 102A-N. For example, clients 102A-N can be desktop computers, laptop computers, mobile devices, etc.

[0020] Clients 102A-N can interact with shards 104A-N through a network 106. Network 106

can be, or can include, a local area network (LAN), a wide area network (WAN), and/or the Internet. Communication over network 106 can be achieved through a suite of network communication protocols such as Ethernet, Transmission Control Protocol/Internet Protocol (TCP/IP), Hypertext Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP), Open Database Connectivity (ODBC), etc. Each of clients 102A-N can execute a separate instance of a software program that utilizes a hash function in order to calculate, based on the primary key of a particular data item, the identity of a particular shard, among shards 104A-N, on which that particular data item either has been stored or is to be stored. Such data items can be separate records possessing different values for similar attribute sets. At least in one embodiment, such data items can be stored within shards 104A-N as separate rows within one or more relational database tables. With very specific exceptions discussed below that are applicable to system 100 during rebalancing events, each data item is located on only one of shards 104A-N at any particular moment in time. A rebalancing event can potentially cause various data items to be relocated from one shard to another shard. In an embodiment of the invention, the sequence of activities involved in a client's performance of an operation during a rebalancing event can differ from the sequence of activities involved in that client's performance of that same type of operation during "normal" system states occurring outside of a rebalancing event.

[0021] Generally, in order to perform an operation relative to a data item that already is stored on a particular one of shards 104A-N, a particular client of clients 102A-N can first determine (e.g., based on the hash function) the shard on which that data item is currently stored. During a rebalancing event, the particular client can perform checks to ensure that the particular client will be operating on the correct copy of the data item, to compensate for the possibility that the data item might have been relocated or deleted asynchronously to the particular client's activities. Although during normal system states only one copy of a data item can exist anywhere in system 100, during a rebalancing event it is possible for multiple copies—different versions—of a data item to be present temporarily within system 100. In performing these checks, the particular client can make use of version information (discussed in greater detail below) that is stored with each copy of each data item. Based at least in part on such version information, the particular client can ensure that the operation, if performed, will be performed relative to the copy of the data item stored on the shard on which the data item was most recently placed. The particular client's performance of the operation during a rebalancing event can involve the creation of a

new copy of the data item on a shard separate from the shard on which another copy of the data item previously existed. In an embodiment, the operation can involve the execution of one or more instructions relative to the data item. For example, such instructions can take the form of query language instructions—Structured Query Language (SQL) instructions being just one specific possibility. After the particular client's performance of the operation relative to a newest copy of the data item, a cleanup operation can be performed to remove old and outdated versions of the data item from shards on which the data item should no longer exist. Generally, data consistency within system 100 can be maintained in this manner without the use of locks by clients 102A-N.

[0022] According to an embodiment of the invention, system 100 is scalable because shards can be added to (or removed from) system 100. The modification of the quantity of shards 104A-N can cause a rebalancing event to occur within system 100. During the rebalancing event, rebalancing processes can re-hash each data item's primary key based on the new quantity of shards instead of the old quantity, thereby determining the identity of the shard (different or the same) on which that data item should be located as of the conclusion of the rebalancing event. The rebalancing processes can then relocate data items from shard to shard using techniques discussed in greater detail below. The relocation can involve the creation and deletion of copies of the data items that would not exist within system 100 outside of the rebalancing event. The rebalancing processes can execute asynchronously to software executing on clients 102A-N. Beneficially, clients 102A-N can continue to perform operations relative to data items during the rebalancing operation, at least in part by adjusting the sequence of activities that clients 102A-N perform during the rebalancing operation.

VERSIONS AND TOMBSTONE ATTRIBUTES

[0023] As is mentioned above, in an embodiment of the invention, each copy of a particular data item can be stored in association with version information. In an embodiment, such version information can take the form of a system-wide version number that is incremented each time that rebalancing event occurs in system 100. Thus, for example, if an existing data item's version number is 3, and if the system's current version number is 4 at the time a new copy of the data item is created on a different shard during a rebalancing event, then the new copy of that

data item will have version number 4. By examining the version information of two separate copies of a data item, which may exist during a rebalancing event, a client can determine which copy is the newer version, and can perform operations relative to that copy. Additionally, by examining the version information of two separate copies of a data item, which may exist during a rebalancing event, a rebalancing process can determine which copy is the older version, and therefore ought to be removed from the shard on which that copy is located.

[0024] Operations that a client performs can involve deleting a data item from a shard. In an embodiment, a client's performance of a deletion operation relative to a data item does not instantly remove all traces of that data item from the shard on which it was located. Instead, in an embodiment, each data item has an attribute called a "tombstone" whose value can be set to true (if that copy of the data item on that shard has been deleted) or false (if that copy of the data item on that shard has not been deleted). The setting of a data item's tombstone attribute to "true" avoids ambiguous situations that otherwise might occur when a data item's absence on a shard could be due either to deletion or to movement to another shard as part of a rebalancing event. In an embodiment, only rebalancing processes (and not clients) are permitted to remove data items from shards completely.

SYSTEM STATE TRANSITIONS

[0025] As is discussed above, at different moments of time, system 100 could be in a state in which it is performing a rebalancing event, or system 100 could be in a state outside of such a rebalancing event. Also as is discussed above, the sequences of activities that clients 102A-N perform as part of operations relative to data items during a rebalancing event can differ from the sequences of activities that clients 102A-N perform as parts of operations of the same type during states in which an rebalancing event is not occurring. Viewed in a simplified manner, during the "normal" state that exists while system 100 is not undergoing a rebalancing event, clients 102A-N may perform operations relative to data items using simple, highly efficient "naïve" techniques that may lack safeguards that protect against certain kinds of inconsistencies. In contrast, during a state that exists while system 100 is undergoing a rebalancing event, clients 102A-N may perform the same types of operations using more complex, more cautious techniques that impose such safeguards. Such safeguards, which might be unnecessary outside

of rebalancing events, can be suitable during rebalancing events.

[0026] FIG. 2 is a state diagram that illustrates the various states 200 in which clients 102A-N can exist at various moments in time, and the possible transitions between those states, according to an embodiment of the invention. States 200 can include a normal state 202, an enter rebalance state 204, a rebalancing state 206, an enter cleanup state 208, a cleanup state 210, and a leave rebalance state 212. In one embodiment, clients 102A-N are permitted to perform operations relative to data items stored within shards 104A-N only during normal state 202 and rebalancing state 206, which are likely to be the states in which clients 102A-N are during the vast majority of the time. Although clients 102A-N will usually be in the same state, separate ones of those clients can briefly be in different states during state transitions, as will be seen from the discussion below.

[0027] According to an embodiment, clients 102A-N can initialize in normal state 202. Clients 102A-N can begin new operations while in normal state 202. A notification mechanism can inform each of clients 102A-N that one or more shards have been added to or are going to be removed from system 100. In response to such a notification, each of clients 102A-N can wait for its pending operations to complete, and then that client can transition to enter rebalance state 204. Clients 102A-N do not begin new operations while in enter rebalance state 204; clients 102A-N can queue up operations to be performed. When all of clients 102A-N have transitioned to enter rebalance state 204, each of clients 102A-N can transition to rebalancing state 206. Once clients 102A-N have entered rebalancing state 206, rebalancing processes can proceed to move data items from source shards to destination shards. In an embodiment of the invention, the current version number, with which new data item copies will become associated from that moment onward, can be incremented upon the entry of clients 102A-N into rebalancing state 206. Clients 102A-N can begin new operations (potentially including queued up operations) while in rebalancing state 206. When the rebalancing processes have moved all data items that are to be relocated, a notification mechanism can inform each of clients 102A-N of this fact. In response to such a notification, each of clients 102A-N can wait for its pending operations to complete, and then that client can transition to enter cleanup state 208. Clients 102A-N do not begin new operations while in enter cleanup state 208; clients 102A-N can queue up operations to be performed. When all of clients 102A-N have transitioned to enter cleanup state 208, each

of clients 102A-N can transition to cleanup state 210. Once clients 102A-N have entered cleanup state 210, rebalancing processes can proceed to remove, from the shards, data item copies that should no longer exist on any shard. In an embodiment, while clients 102A-N are in cleanup state 210, rebalancing processes can remove, from the shards, all data item copies having a "true" tombstone attribute value. Additionally, in an embodiment, while clients 102A-N are in cleanup state 210, rebalancing processes can remove, from the shards, all data item copies having a version number attribute value that is less than the system's current version number. Clients 102A-N do not begin new operations while in cleanup state 210; clients 102A-N can queue up operations to be performed. When the rebalancing processes have removed all data item copies that are to be removed, a notification mechanism can inform each of clients 102A-N of this fact. In response to such a notification, each of clients 102A-N can transition to leave rebalance state 212. Clients 102A-N do not begin new operations while in leave rebalance state 212; clients 102A-N can queue up operations to be performed. When all of clients 102A-N have transitioned to leave rebalance state 212, each of clients 102A-N can transition back to normal state 202. Clients 102A-N can begin new operations (potentially including queued up operations) while in normal state 202.

[0028] As is discussed above, while clients 102A-N are in rebalancing state 206, clients 102A-N can perform operations in a more cautious manner than then manner in which clients 102A-N would perform the same types of operations while in normal state 202. The manner in which clients 102A-N can perform operations while in rebalancing state 206 can guarantee data consistency in spite of the concurrent asynchronous execution of rebalancing processes that may be relocating data items from shard to shard—a concern that does not exist in normal state 202. Discussed below are techniques for performing various different types of operations in this more cautious, consistency-guaranteeing manner while in rebalancing state 206.

ADD OPERATIONS

[0029] FIG. 3 is a flow diagram that illustrates an example of a technique 300 for performing an add operation while in the rebalancing state, according to an embodiment of the invention. Although technique 300 is illustrated as including specific activities performed in a specific order, alternative embodiments of the invention can involve techniques having additional, fewer,

or different activities. Any of clients 102A-N can perform technique 300. In block 302, a client can determine the identity of the source shard on which a particular data item would have been located prior to the change in shard quantity. This determination can be achieved, for example, by calculating the particular data item's primary key modulo the old shard quantity. In block 304, the client can determine the identity of the destination shard on which the particular data item is to be located following the change in shard quantity. This determination can be achieved, for example, by calculating the particular data item's primary key modulo the new shard quantity. In block 306, the client can determine whether the identity of the source shard is the same as the identity of the destination shard. If the identities are the same, then control passes to block 308. Otherwise, control passes to block 310.

[0030] In block 308, the client can add the particular data item to the destination shard in the normal, naïve, highly efficient standard manner for performing an add operation. At this point, technique 300 terminates.

[0031] Alternatively, in block 310, the client can determine whether a data item having the particular data item's primary key already exists on the source shard. If a data item having the particular data item's primary key already exists on the source shard, then control passes to block 312. Otherwise, control passes to block 314.

[0032] In block 312, the client can conclude that the particular data item duplicates a data item already existing in the shard system, and can refrain from performing the add operation. The client can signify to a user that the add operation was prevented due to duplication. At this point, technique 300 terminates.

[0033] Alternatively, in block 314, the client can determine whether there already exists, on the destination shard, a data item having both (a) the particular data item's primary key and (b) a tombstone attribute value of "true." If there already exists, on the destination shard, a data item having both (a) the particular data item's primary key and (b) a tombstone attribute value of "true," then control passes to block 316. Otherwise, control passes to block 320.

[0034] In block 316, as part of an atomic action, the client can assign, to the attribute values of the data item having the particular data item's primary key (on the destination shard), the attribute values of the particular data item. This assignment essentially updates the data item

existing on the destination shard. The client can assign the system's current version number to the data item's version number attribute. In block 318, as part of the same atomic action, the client can set the tombstone attribute value of the data item having the particular data item's primary key (on the destination shard) to "false." In an embodiment, the attribute value assignment of block 316 achieves the same result as that achieved by the activity of block 318, since the particular data item's tombstone attribute value will already be "false" prior to the assignment. The client can signify to a user that the add operation succeeded. At this point, technique 300 terminates.

[0035] Alternatively, in block 320, the client can insert the particular data item into the destination shard. The client can assign the system's current version number to the particular data item's version number attribute. The client can signify to a user that the add operation succeeded. At this point, technique 300 terminates. In an embodiment of the invention, the activities of blocks 314-320 are performed as a single atomic operation.

UPDATE OPERATIONS

[0036] FIG. 4 is a flow diagram that illustrates an example of a technique 400 for performing an update operation while in the rebalancing state, according to an embodiment of the invention. Although technique 400 is illustrated as including specific activities performed in a specific order, alternative embodiments of the invention can involve techniques having additional, fewer, or different activities. Any of clients 102A-N can perform technique 400. In block 402, a client can determine the identity of the source shard on which a particular data item would have been located prior to the change in shard quantity. This determination can be achieved, for example, by calculating the particular data item's primary key modulo the old shard quantity. In block 404, the client can determine the identity of the destination shard on which the particular data item is to be located following the change in shard quantity. This determination can be achieved, for example, by calculating the particular data item's primary key modulo the new shard quantity. In block 406, the client can determine whether the identity of the source shard is the same as the identity of the destination shard. If the identities are the same, then control passes to block 408. Otherwise, control passes to block 410.

[0037] In block 408, the client can update the particular data item on the destination shard in the normal, naïve, highly efficient standard manner for performing an update operation. At this point, technique 400 terminates.

[0038] Alternatively, in block 410, the client can determine whether a data item having the particular data item's primary key already exists on the source shard. If a data item having the particular data item's primary key already exists on the source shard, then control passes to block 416. Otherwise, control passes to block 422.

[0039] In block 416, the client can determine whether there already exists, on the destination shard, a data item having the particular data item's primary key. If there already exists, on the destination shard, a data item having the particular data item's primary key, then control passes to block 418. Otherwise, control passes to block 420.

[0040] In block 418, the client can assign, to the attribute values of the data item having the particular data item's primary key (on the destination shard), the attribute values of the particular data item. This assignment essentially updates the data item existing on the destination shard. The client can assign the system's current version number to the data item's version number attribute. The client can signify to a user that the update operation succeeded. At this point, technique 400 terminates.

[0041] Alternatively, in block 420, the client can insert the particular data item into the destination shard. The client can assign the system's current version number to the particular data item's version number attribute. The client can signify to a user that the update operation succeeded. At this point, technique 400 terminates.

[0042] Alternatively, in block 422, the client can determine whether there already exists, on the destination shard, a data item having both (a) the particular data item's primary key and (b) a tombstone attribute value of "false." If there already exists, on the destination shard, a data item having both (a) the particular data item's primary key and (b) a tombstone attribute value of "false," then control passes to block 418. Otherwise, control passes to block 428.

[0043] Alternatively, in block 428, the client can refrain from performing the update operation. The client can signify to a user that the update operation failed (because there was no data item to update). At this point, technique 400 terminates.

DELETE OPERATIONS

[0044] FIG. 5 is a flow diagram that illustrates an example of a technique 500 for performing a delete operation while in the rebalancing state, according to an embodiment of the invention. Although technique 500 is illustrated as including specific activities performed in a specific order, alternative embodiments of the invention can involve techniques having additional, fewer, or different activities. Any of clients 102A-N can perform technique 500. In block 502, a client can determine the identity of the source shard on which a particular data item would have been located prior to the change in shard quantity. This determination can be achieved, for example, by calculating the particular data item's primary key modulo the old shard quantity. In block 504, the client can determine the identity of the destination shard on which the particular data item is to be located following the change in shard quantity. This determination can be achieved, for example, by calculating the particular data item's primary key modulo the new shard quantity. In block 506, the client can determine whether the identity of the source shard is the same as the identity of the destination shard. If the identities are the same, then control passes to block 508. Otherwise, control passes to block 510.

[0045] In block 508, the client can delete the particular data item on the destination shard in the normal, naïve, highly efficient standard manner for performing a delete operation. At this point, technique 500 terminates.

[0046] Alternatively, in block 510, as part of an atomic action, the client can determine whether a data item having the particular data item's primary key already exists on the source shard. If a data item having the particular data item's primary key already exists on the source shard, then control passes to block 512. Otherwise, control passes to block 516.

[0047] In block 512, as part of the atomic action, the client can upsert the particular data item into the destination shard. An upsert is defined as: (1) an insert if the object identified by the primary key does not exist in the database (i.e., the destination shard), or (2) an update if the object identified by the primary key does exist in the database (i.e., the destination shard). In block 514, as part of the atomic action, the client can set the value of the particular data item's tombstone attribute to "true." The client can assign the system's current version number to the

particular data item's version number attribute. The client can signify to a user that the delete operation succeeded. At this point, technique 500 terminates.

[0048] Alternatively, in block 516, the client can determine whether a data item having the particular data item's primary key already exists on the destination shard. If a data item having the particular data item's primary key already exists on the destination shard, then control passes to block 518. Otherwise, control passes to block 520.

[0049] In block 518, the client can set the value of the data item's tombstone attribute to "true" (on the destination shard). The client can assign the system's current version number to the data item's version number attribute. The client can signify to a user that the delete operation succeeded. At this point, technique 500 terminates.

[0050] Alternatively, in block 520, the client can refrain from performing the delete operation. The client can signify to a user that the delete operation failed (because there was no data item to delete). At this point, technique 500 terminates.

GET OPERATIONS

[0051] FIGs. 6A-6B are flow diagrams that illustrate an example of a technique 600 for performing a get operation while in the rebalancing state, according to an embodiment of the invention. In an embodiment, the get operation can read and return the attribute values of a data item having a client-specified primary key. Although technique 600 is illustrated as including specific activities performed in a specific order, alternative embodiments of the invention can involve techniques having additional, fewer, or different activities. Any of clients 102A-N can perform technique 600. Referring first to FIG. 6A, in block 602, a client can determine the identity of the source shard on which a particular data item having a specified primary key would have been located prior to the change in shard quantity. This determination can be achieved, for example, by calculating the specified primary key modulo the old shard quantity. In block 604, the client can determine the identity of the destination shard on which the particular data item having the specified primary key is to be located following the change in shard quantity. This determination can be achieved, for example, by calculating the specified primary key modulo the new shard quantity. In block 606, the client can determine whether the identity of the source

shard is the same as the identity of the destination shard. If the identities are the same, then control passes to block 608. Otherwise, control passes to block 610.

[0052] In block 608, the client can perform a get operation relative to a particular data item having the specified primary key on the destination shard in the normal, naïve, highly efficient standard manner for performing a get operation. At this point, technique 600 terminates.

[0053] Alternatively, in block 610, the client can determine whether a data item having the specified primary key already exists on the destination shard. If a data item having the specified primary key already exists on the destination shard, then control passes to block 612. Otherwise, control passes to block 618.

[0054] In block 612, the client can determine whether the data item having the specified primary key on the destination shard has a "true" tombstone attribute value. If the data item having the specified primary key on the destination shard has a "true" tombstone attribute value, then control passes to block 614. Otherwise, control passes to block 616.

[0055] In block 614, the client can refrain from performing the get operation. The client can signify to a user that the get operation failed (because no data item having the specified primary key was found). At this point, technique 600 terminates.

[0056] Alternatively, in block 616, the client can read the attribute values of the data having the specified primary key on the destination shard. The client can present these attribute values to a user, coincidentally signifying that the get operation succeeded. At this point, technique 600 terminates.

[0057] Alternatively, in block 618, the client can determine whether a data item having the specified primary key already exists on the source shard. If a data item having the specified primary key already exists on the source shard, then control passes to block 620. Otherwise, control passes to block 626 on FIG. 6B.

[0058] Alternatively, in block 620, the client can determine whether the data item having the specified primary key on the source shard has a "true" tombstone attribute value. If the data item having the specified primary key on the source shard has a "true" tombstone attribute value, then control passes to block 622. Otherwise, control passes to block 624.

[0059] In block 622, the client can refrain from performing the get operation. The client can signify to a user that the get operation failed (because no data item having the specified primary key was found). At this point, technique 600 terminates.

[0060] Alternatively, in block 624, the client can read the attribute values of the data having the specified primary key on the source shard. The client can present these attribute values to a user, coincidentally signifying that the get operation succeeded. At this point, technique 600 terminates.

[0061] Referring now to FIG. 6B, alternatively, in block 626, the client can determine (again) whether a data item having the specified primary key now exists on the destination shard. If a data item having the specified primary key now exists on the destination shard (e.g., as a consequence of a rebalancing process having recently moved that data item to the destination shard), then control passes to block 628. Otherwise, control passes to block 630.

[0062] In block 628, the client can determine whether the data item having the specified primary key on the destination shard has a "true" tombstone attribute value. If the data item having the specified primary key on the destination shard has a "true" tombstone attribute value, then control passes to block 630. Otherwise, control passes to block 632.

[0063] In block 630, the client can refrain from performing the get operation. The client can signify to a user that the get operation failed (because no data item having the specified primary key was found). At this point, technique 600 terminates.

[0064] Alternatively, in block 632, the client can read the attribute values of the data having the specified primary key on the destination shard. The client can present these attribute values to a user, coincidentally signifying that the get operation succeeded. At this point, technique 600 terminates.

REBALANCING OPERATIONS

[0065] FIG. 7 is a flow diagram that illustrates an example of a technique 700 for performing a rebalancing operation while clients 102A-N are in the rebalancing state (initially), according to an embodiment of the invention. Although technique 700 is illustrated as including specific

activities performed in a specific order, alternative embodiments of the invention can involve techniques having additional, fewer, or different activities. Rebalancing processes can perform technique 700 asynchronously to the performance of other types of operations by clients 102A-N.

[0066] In block 702, the system's current version number is incremented. In block 704, a rebalancing process can determine whether any shard still contains a data item whose version number attribute value is less than the system's current version number. If a shard contains a particular data item whose version number attribute value is less than the system's current version number, then control passes to block 706. Otherwise, the rebalancing processes have finished relocating data items for this particular rebalancing event, and control passes to block 714.

[0067] In block 706, the rebalancing process can determine the identity of the destination shard on which the particular data item is to be located following the change in shard quantity. This determination can be achieved, for example, by calculating the particular data item's primary key modulo the new shard quantity. In block 708, the rebalancing process can determine whether the identity of a source shard, on which the particular data item is currently located, is the same as the identity of the destination shard. If the identities are the same, then control passes to block 710. Otherwise, control passes to block 712.

[0068] In block 710, the rebalancing process can assign the system's current version number to the particular data item's version number attribute. Under such circumstances, the particular data item does not need to be relocated to a different shard. Control passes back to block 704.

[0069] Alternatively, in block 712, the rebalancing process can insert the particular data item into the destination shard. In one embodiment, potential conflicts can be ignored. The rebalancing process can assign the system's current version number to the particular data item's version number attribute. Significantly, in an embodiment, an old copy of the particular data item can remain on the source shard until the activities of blocks 716 and 718 are performed. Control passes back to block 704.

[0070] Alternatively, in block 714, the rebalancing process can determine whether any queries (e.g., from clients 102A-N) that had been executing as of the time that the rebalancing processes

finished relocating data items are currently executing against any of the shards (e.g., shards 104A-N). If at least one query that had been executing as of the time that the rebalancing processes finished relocating data items is currently executing against at least one shard, then control passes back to block 714. Otherwise, control passes to block 716.

[0071] After all queries that were pending as of the conclusion of the data item relocation have completed, in block 716, the rebalancing processes can remove, from all shards, all data item copies whose version number attribute value is less than the system's current version number. In block 718, the rebalancing processes can remove, from all shards, all data item copies whose tombstone attribute values are "true." In an embodiment, the activities of blocks 716 and 718 can be performed during cleanup phase 210 discussed above in connection with FIG. 2.

QUERY OPERATIONS

[0072] FIG. 8 is a flow diagram that illustrates an example of a technique 800 for performing a query operation while in the rebalancing state, according to an embodiment of the invention. Although technique 800 is illustrated as including specific activities performed in a specific order, alternative embodiments of the invention can involve techniques having additional, fewer, or different activities. Any of clients 102A-N can perform technique 800, for example. Technique 800 can be performed concurrently with technique 700. In block 802, for each particular shard of shards 104A-N, all of the particular shard's data items that satisfy query-specified filtering criteria can be placed in a separate preliminary result queue for that particular shard. Each such preliminary result queue can start out empty. Thus, a separate preliminary result queue may be populated for each of shards 104A-N. In block 804, for each particular shard of shards 104A-N, the data items contained within that particular shard's preliminary result queue can be sorted based at least in part on those data items' primary keys. As a result, for example, each preliminary result queue can contain data items that are sorted such that the data item having the smallest primary key of data items in that preliminary result queue can be at the front of that preliminary result queue. In block 806, a determination can be made as to whether all of the shards' preliminary result queues are empty. If all of the shards' preliminary result queues are empty, then control passes to block 818. Otherwise, control passes to block 808.

[0073] In block 808, from the set of data items that are currently at the top of each shard's preliminary result queue, a subset of one or more data items having the smallest primary key among data items in that set can be selected. In block 810, from the subset of one or more data items selected in block 808, a particular data item having the largest version number attribute value can be selected. In block 812, a determination can be made as to whether the particular data item's tombstone attribute value is "false." If the particular data item's tombstone attribute value is "false," then control passes to block 814. Otherwise, control passes to block 816.

[0074] In block 814, the particular data item can be added to a final result set. The final result set can start out empty. Control passes to block 816. In block 816, all data item copies having the same primary key as the particular data item (including the particular data item itself) can be removed from all of the shards' preliminary result queues. This removal potentially can cause other data items to rise to the top of one or more of those queues. Control passes back to block 806.

[0075] Alternatively, in block 818, the data items in the final result set can be returned as the final results of the query operation.

HARDWARE OVERVIEW

[0076] FIG. 9 depicts a simplified diagram of a distributed system 900 for implementing one of the embodiments. In the illustrated embodiment, distributed system 900 includes one or more client computing devices 902, 904, 906, and 908, which are configured to execute and operate a client application such as a web browser, proprietary client (e.g., Oracle Forms), or the like over one or more network(s) 910. Server 912 may be communicatively coupled with remote client computing devices 902, 904, 906, and 908 via network 910.

[0077] In various embodiments, server 912 may be adapted to run one or more services or software applications provided by one or more of the components of the system. In some embodiments, these services may be offered as web-based or cloud services or under a Software as a Service (SaaS) model to the users of client computing devices 902, 904, 906, and/or 908. Users operating client computing devices 902, 904, 906, and/or 908 may in turn utilize one or

more client applications to interact with server 912 to utilize the services provided by these components.

[0078] In the configuration depicted in the figure, the software components 918, 920 and 922 of system 900 are shown as being implemented on server 912. In other embodiments, one or more of the components of system 900 and/or the services provided by these components may also be implemented by one or more of the client computing devices 902, 904, 906, and/or 908. Users operating the client computing devices may then utilize one or more client applications to use the services provided by these components. These components may be implemented in hardware, firmware, software, or combinations thereof. It should be appreciated that various different system configurations are possible, which may be different from distributed system 900. The embodiment shown in the figure is thus one example of a distributed system for implementing an embodiment system and is not intended to be limiting.

[0079] Client computing devices 902, 904, 906, and/or 908 may be portable handheld devices (e.g., an iPhone®, cellular telephone, an iPad®, computing tablet, a personal digital assistant (PDA)) or wearable devices (e.g., a Google Glass® head mounted display), running software such as Microsoft Windows Mobile®, and/or a variety of mobile operating systems such as iOS, Windows Phone, Android, BlackBerry 10, Palm OS, and the like, and being Internet, e-mail, short message service (SMS), Blackberry®, or other communication protocol enabled. The client computing devices can be general purpose personal computers including, by way of example, personal computers and/or laptop computers running various versions of Microsoft Windows®, Apple Macintosh®, and/or Linux operating systems. The client computing devices can be workstation computers running any of a variety of commercially-available UNIX® or UNIX-like operating systems, including without limitation the variety of GNU/Linux operating systems, such as for example, Google Chrome OS. Alternatively, or in addition, client computing devices 902, 904, 906, and 908 may be any other electronic device, such as a thin-client computer, an Internet-enabled gaming system (e.g., a Microsoft Xbox gaming console with or without a Kinect® gesture input device), and/or a personal messaging device, capable of communicating over network(s) 910.

[0080] Although exemplary distributed system 900 is shown with four client computing devices, any number of client computing devices may be supported. Other devices, such as devices with sensors, etc., may interact with server 912.

[0081] Network(s) 910 in distributed system 900 may be any type of network familiar to those skilled in the art that can support data communications using any of a variety of commercially-available protocols, including without limitation TCP/IP (transmission control protocol/Internet protocol), SNA (systems network architecture), IPX (Internet packet exchange), AppleTalk, and the like. Merely by way of example, network(s) 910 can be a local area network (LAN), such as one based on Ethernet, Token-Ring and/or the like. Network(s) 910 can be a wide-area network and the Internet. It can include a virtual network, including without limitation a virtual private network (VPN), an intranet, an extranet, a public switched telephone network (PSTN), an infra-red network, a wireless network (e.g., a network operating under any of the Institute of Electrical and Electronics (IEEE) 902.11 suite of protocols, Bluetooth®, and/or any other wireless protocol); and/or any combination of these and/or other networks.

[0082] Server 912 may be composed of one or more general purpose computers, specialized server computers (including, by way of example, PC (personal computer) servers, UNIX® servers, mid-range servers, mainframe computers, rack-mounted servers, etc.), server farms, server clusters, or any other appropriate arrangement and/or combination. In various embodiments, server 912 may be adapted to run one or more services or software applications described in the foregoing disclosure. For example, server 912 may correspond to a server for performing processing described above according to an embodiment of the present disclosure.

[0083] Server 912 may run an operating system including any of those discussed above, as well as any commercially available server operating system. Server 912 may also run any of a variety of additional server applications and/or mid-tier applications, including HTTP (hypertext transport protocol) servers, FTP (file transfer protocol) servers, CGI (common gateway interface) servers, JAVA® servers, database servers, and the like. Exemplary database servers include without limitation those commercially available from Oracle, Microsoft, Sybase, IBM (International Business Machines), and the like.

[0084] In some implementations, server 912 may include one or more applications to analyze and consolidate data feeds and/or event updates received from users of client computing devices 902, 904, 906, and 908. As an example, data feeds and/or event updates may include, but are not limited to, Twitter® feeds, Facebook® updates or real-time updates received from one or more third party information sources and continuous data streams, which may include real-time events related to sensor data applications, financial tickers, network performance measuring tools (e.g.,

network monitoring and traffic management applications), clickstream analysis tools, automobile traffic monitoring, and the like. Server 912 may also include one or more applications to display the data feeds and/or real-time events via one or more display devices of client computing devices 902, 904, 906, and 908.

[0085] Distributed system 900 may also include one or more databases 914 and 916. Databases 914 and 916 may reside in a variety of locations. By way of example, one or more of databases 914 and 916 may reside on a non-transitory storage medium local to (and/or resident in) server 912. Alternatively, databases 914 and 916 may be remote from server 912 and in communication with server 912 via a network-based or dedicated connection. In one set of embodiments, databases 914 and 916 may reside in a storage-area network (SAN). Similarly, any necessary files for performing the functions attributed to server 912 may be stored locally on server 912 and/or remotely, as appropriate. In one set of embodiments, databases 914 and 916 may include relational databases, such as databases provided by Oracle, which are adapted to store, update, and retrieve data in response to SQL-formatted commands.

[0086] FIG. 10 is a simplified block diagram of one or more components of a system environment 1000 by which services provided by one or more components of an embodiment system may be offered as cloud services, in accordance with an embodiment of the present disclosure. In the illustrated embodiment, system environment 1000 includes one or more client computing devices 1004, 1006, and 1008 that may be used by users to interact with a cloud infrastructure system 1002 that provides cloud services. The client computing devices may be configured to operate a client application such as a web browser, a proprietary client application (e.g., Oracle Forms), or some other application, which may be used by a user of the client computing device to interact with cloud infrastructure system 1002 to use services provided by cloud infrastructure system 1002.

[0087] It should be appreciated that cloud infrastructure system 1002 depicted in the figure may have other components than those depicted. Further, the embodiment shown in the figure is only one example of a cloud infrastructure system that may incorporate an embodiment of the invention. In some other embodiments, cloud infrastructure system 1002 may have more or fewer components than shown in the figure, may combine two or more components, or may have a different configuration or arrangement of components.

[0088] Client computing devices 1004, 1006, and 1008 may be devices similar to those described above for 902, 904, 906, and 908.

[0089] Although exemplary system environment 1000 is shown with three client computing devices, any number of client computing devices may be supported. Other devices such as devices with sensors, etc. may interact with cloud infrastructure system 1002.

[0090] Network(s) 1010 may facilitate communications and exchange of data between clients 1004, 1006, and 1008 and cloud infrastructure system 1002. Each network may be any type of network familiar to those skilled in the art that can support data communications using any of a variety of commercially-available protocols, including those described above for network(s) 910.

[0091] Cloud infrastructure system 1002 may comprise one or more computers and/or servers that may include those described above for server 912.

[0092] In certain embodiments, services provided by the cloud infrastructure system may include a host of services that are made available to users of the cloud infrastructure system on demand, such as online data storage and backup solutions, Web-based e-mail services, hosted office suites and document collaboration services, database processing, managed technical support services, and the like. Services provided by the cloud infrastructure system can dynamically scale to meet the needs of its users. A specific instantiation of a service provided by cloud infrastructure system is referred to herein as a "service instance." In general, any service made available to a user via a communication network, such as the Internet, from a cloud service provider's system is referred to as a "cloud service." Typically, in a public cloud environment, servers and systems that make up the cloud service provider's system are different from the customer's own on-premises servers and systems. For example, a cloud service provider's system may host an application, and a user may, via a communication network such as the Internet, on demand, order and use the application.

[0093] In some examples, a service in a computer network cloud infrastructure may include protected computer network access to storage, a hosted database, a hosted web server, a software application, or other service provided by a cloud vendor to a user, or as otherwise known in the art. For example, a service can include password-protected access to remote storage on the cloud through the Internet. As another example, a service can include a web service-based hosted relational database and a script-language middleware engine for private use by a networked

developer. As another example, a service can include access to an email software application hosted on a cloud vendor's web site.

[0094] In certain embodiments, cloud infrastructure system 1002 may include a suite of applications, middleware, and database service offerings that are delivered to a customer in a self-service, subscription-based, elastically scalable, reliable, highly available, and secure manner. An example of such a cloud infrastructure system is the Oracle Public Cloud provided by the present assignee.

[0095] In various embodiments, cloud infrastructure system 1002 may be adapted to automatically provision, manage and track a customer's subscription to services offered by cloud infrastructure system 1002. Cloud infrastructure system 1002 may provide the cloud services via different deployment models. For example, services may be provided under a public cloud model in which cloud infrastructure system 1002 is owned by an organization selling cloud services (e.g., owned by Oracle) and the services are made available to the general public or different industry enterprises. As another example, services may be provided under a private cloud model in which cloud infrastructure system 1002 is operated solely for a single organization and may provide services for one or more entities within the organization. The cloud services may also be provided under a community cloud model in which cloud infrastructure system 1002 and the services provided by cloud infrastructure system 1002 are shared by several organizations in a related community. The cloud services may also be provided under a hybrid cloud model, which is a combination of two or more different models.

[0096] In some embodiments, the services provided by cloud infrastructure system 1002 may include one or more services provided under Software as a Service (SaaS) category, Platform as a Service (PaaS) category, Infrastructure as a Service (IaaS) category, or other categories of services including hybrid services. A customer, via a subscription order, may order one or more services provided by cloud infrastructure system 1002. Cloud infrastructure system 1002 then performs processing to provide the services in the customer's subscription order.

[0097] In some embodiments, the services provided by cloud infrastructure system 1002 may include, without limitation, application services, platform services and infrastructure services. In some examples, application services may be provided by the cloud infrastructure system via a SaaS platform. The SaaS platform may be configured to provide cloud services that fall under the SaaS category. For example, the SaaS platform may provide capabilities to build and deliver

a suite of on-demand applications on an integrated development and deployment platform. The SaaS platform may manage and control the underlying software and infrastructure for providing the SaaS services. By utilizing the services provided by the SaaS platform, customers can utilize applications executing on the cloud infrastructure system. Customers can acquire the application services without the need for customers to purchase separate licenses and support. Various different SaaS services may be provided. Examples include, without limitation, services that provide solutions for sales performance management, enterprise integration, and business flexibility for large organizations.

[0098] In some embodiments, platform services may be provided by the cloud infrastructure system via a PaaS platform. The PaaS platform may be configured to provide cloud services that fall under the PaaS category. Examples of platform services may include without limitation services that enable organizations (such as Oracle) to consolidate existing applications on a shared, common architecture, as well as the ability to build new applications that leverage the shared services provided by the platform. The PaaS platform may manage and control the underlying software and infrastructure for providing the PaaS services. Customers can acquire the PaaS services provided by the cloud infrastructure system without the need for customers to purchase separate licenses and support. Examples of platform services include, without limitation, Oracle Java Cloud Service (JCS), Oracle Database Cloud Service (DBCS), and others.

[0099] By utilizing the services provided by the PaaS platform, customers can employ programming languages and tools supported by the cloud infrastructure system and also control the deployed services. In some embodiments, platform services provided by the cloud infrastructure system may include database cloud services, middleware cloud services (e.g., Oracle Fusion Middleware services), and Java cloud services. In one embodiment, database cloud services may support shared service deployment models that enable organizations to pool database resources and offer customers a Database as a Service in the form of a database cloud. Middleware cloud services may provide a platform for customers to develop and deploy various business applications, and Java cloud services may provide a platform for customers to deploy Java applications, in the cloud infrastructure system.

[0100] Various different infrastructure services may be provided by an IaaS platform in the cloud infrastructure system. The infrastructure services facilitate the management and control of the underlying computing resources, such as storage, networks, and other fundamental

computing resources for customers utilizing services provided by the SaaS platform and the PaaS platform.

[0101] In certain embodiments, cloud infrastructure system 1002 may also include infrastructure resources 1030 for providing the resources used to provide various services to customers of the cloud infrastructure system. In one embodiment, infrastructure resources 1030 may include pre-integrated and optimized combinations of hardware, such as servers, storage, and networking resources to execute the services provided by the PaaS platform and the SaaS platform.

[0102] In some embodiments, resources in cloud infrastructure system 1002 may be shared by multiple users and dynamically re-allocated per demand. Additionally, resources may be allocated to users in different time zones. For example, cloud infrastructure system 1030 may enable a first set of users in a first time zone to utilize resources of the cloud infrastructure system for a specified number of hours and then enable the re-allocation of the same resources to another set of users located in a different time zone, thereby maximizing the utilization of resources.

[0103] In certain embodiments, a number of internal shared services 1032 may be provided that are shared by different components or modules of cloud infrastructure system 1002 and by the services provided by cloud infrastructure system 1002. These internal shared services may include, without limitation, a security and identity service, an integration service, an enterprise repository service, an enterprise manager service, a virus scanning and white list service, a high availability, backup and recovery service, service for enabling cloud support, an email service, a notification service, a file transfer service, and the like.

[0104] In certain embodiments, cloud infrastructure system 1002 may provide comprehensive management of cloud services (e.g., SaaS, PaaS, and IaaS services) in the cloud infrastructure system. In one embodiment, cloud management functionality may include capabilities for provisioning, managing and tracking a customer's subscription received by cloud infrastructure system 1002, and the like.

[0105] In one embodiment, as depicted in the figure, cloud management functionality may be provided by one or more modules, such as an order management module 1020, an order orchestration module 1022, an order provisioning module 1024, an order management and monitoring module 1026, and an identity management module 1028. These modules may

include or be provided using one or more computers and/or servers, which may be general purpose computers, specialized server computers, server farms, server clusters, or any other appropriate arrangement and/or combination.

[0106] In exemplary operation 1034, a customer using a client device, such as client device 1004, 1006 or 1008, may interact with cloud infrastructure system 1002 by requesting one or more services provided by cloud infrastructure system 1002 and placing an order for a subscription for one or more services offered by cloud infrastructure system 1002. In certain embodiments, the customer may access a cloud User Interface (UI), cloud UI 1012, cloud UI 1014 and/or cloud UI 1016 and place a subscription order via these UIs. The order information received by cloud infrastructure system 1002 in response to the customer placing an order may include information identifying the customer and one or more services offered by the cloud infrastructure system 1002 that the customer intends to subscribe to.

[0107] After an order has been placed by the customer, the order information is received via the cloud UIs, 1012, 1014 and/or 1016.

[0108] At operation 1036, the order is stored in order database 1018. Order database 1018 can be one of several databases operated by cloud infrastructure system 1018 and operated in conjunction with other system elements.

[0109] At operation 1038, the order information is forwarded to an order management module 1020. In some instances, order management module 1020 may be configured to perform billing and accounting functions related to the order, such as verifying the order, and upon verification, booking the order.

[0110] At operation 1040, information regarding the order is communicated to an order orchestration module 1022. Order orchestration module 1022 may utilize the order information to orchestrate the provisioning of services and resources for the order placed by the customer. In some instances, order orchestration module 1022 may orchestrate the provisioning of resources to support the subscribed services using the services of order provisioning module 1024.

[0111] In certain embodiments, order orchestration module 1022 enables the management of business processes associated with each order and applies business logic to determine whether an order should proceed to provisioning. At operation 1042, upon receiving an order for a new subscription, order orchestration module 1022 sends a request to order provisioning module 1024 to allocate resources and configure those resources needed to fulfill the subscription order.

Order provisioning module 1024 enables the allocation of resources for the services ordered by the customer. Order provisioning module 1024 provides a level of abstraction between the cloud services provided by cloud infrastructure system 1000 and the physical implementation layer that is used to provision the resources for providing the requested services. Order orchestration module 1022 may thus be isolated from implementation details, such as whether or not services and resources are actually provisioned on the fly or pre-provisioned and only allocated/assigned upon request.

[0112] At operation 1044, once the services and resources are provisioned, a notification of the provided service may be sent to customers on client devices 1004, 1006 and/or 1008 by order provisioning module 1024 of cloud infrastructure system 1002.

[0113] At operation 1046, the customer's subscription order may be managed and tracked by an order management and monitoring module 1026. In some instances, order management and monitoring module 1026 may be configured to collect usage statistics for the services in the subscription order, such as the amount of storage used, the amount data transferred, the number of users, and the amount of system up time and system down time.

[0114] In certain embodiments, cloud infrastructure system 1000 may include an identity management module 1028. Identity management module 1028 may be configured to provide identity services, such as access management and authorization services in cloud infrastructure system 1000. In some embodiments, identity management module 1028 may control information about customers who wish to utilize the services provided by cloud infrastructure system 1002. Such information can include information that authenticates the identities of such customers and information that describes which actions those customers are authorized to perform relative to various system resources (e.g., files, directories, applications, communication ports, memory segments, etc.) Identity management module 1028 may also include the management of descriptive information about each customer and about how and by whom that descriptive information can be accessed and modified.

[0115] FIG. 11 illustrates an example computer system 1100 in which various embodiments of the present invention may be implemented. The system 1100 may be used to implement any of the computer systems described above. As shown in the figure, computer system 1100 includes a processing unit 1104 that communicates with a number of peripheral subsystems via a bus subsystem 1102. These peripheral subsystems may include a processing acceleration unit 1106,

an I/O subsystem 1108, a storage subsystem 1118 and a communications subsystem 1124. Storage subsystem 1118 includes tangible computer-readable storage media 1122 and a system memory 1110.

[0116] Bus subsystem 1102 provides a mechanism for letting the various components and subsystems of computer system 1100 communicate with each other as intended. Although bus subsystem 1102 is shown schematically as a single bus, alternative embodiments of the bus subsystem may utilize multiple buses. Bus subsystem 1102 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. For example, such architectures may include an Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus, which can be implemented as a Mezzanine bus manufactured to the IEEE P1386.1 standard.

[0117] Processing unit 1104, which can be implemented as one or more integrated circuits (e.g., a conventional microprocessor or microcontroller), controls the operation of computer system 1100. One or more processors may be included in processing unit 1104. These processors may include single core or multicore processors. In certain embodiments, processing unit 1104 may be implemented as one or more independent processing units 1132 and/or 1134 with single or multicore processors included in each processing unit. In other embodiments, processing unit 1104 may also be implemented as a quad-core processing unit formed by integrating two dual-core processors into a single chip.

[0118] In various embodiments, processing unit 1104 can execute a variety of programs in response to program code and can maintain multiple concurrently executing programs or processes. At any given time, some or all of the program code to be executed can be resident in processor(s) 1104 and/or in storage subsystem 1118. Through suitable programming, processor(s) 1104 can provide various functionalities described above. Computer system 1100 may additionally include a processing acceleration unit 1106, which can include a digital signal processor (DSP), a special-purpose processor, and/or the like.

[0119] I/O subsystem 1108 may include user interface input devices and user interface output devices. User interface input devices may include a keyboard, pointing devices such as a mouse or trackball, a touchpad or touch screen incorporated into a display, a scroll wheel, a click wheel,

a dial, a button, a switch, a keypad, audio input devices with voice command recognition systems, microphones, and other types of input devices. User interface input devices may include, for example, motion sensing and/or gesture recognition devices such as the Microsoft Kinect® motion sensor that enables users to control and interact with an input device, such as the Microsoft Xbox® 360 game controller, through a natural user interface using gestures and spoken commands. User interface input devices may also include eye gesture recognition devices such as the Google Glass® blink detector that detects eye activity (e.g., 'blinking' while taking pictures and/or making a menu selection) from users and transforms the eye gestures as input into an input device (e.g., Google Glass®). Additionally, user interface input devices may include voice recognition sensing devices that enable users to interact with voice recognition systems (e.g., Siri® navigator), through voice commands.

[0120] User interface input devices may also include, without limitation, three dimensional (3D) mice, joysticks or pointing sticks, gamepads and graphic tablets, and audio/visual devices such as speakers, digital cameras, digital camcorders, portable media players, webcams, image scanners, fingerprint scanners, barcode reader 3D scanners, 3D printers, laser rangefinders, and eye gaze tracking devices. Additionally, user interface input devices may include, for example, medical imaging input devices such as computed tomography, magnetic resonance imaging, position emission tomography, medical ultrasonography devices. User interface input devices may also include, for example, audio input devices such as MIDI keyboards, digital musical instruments and the like.

[0121] User interface output devices may include a display subsystem, indicator lights, or non-visual displays such as audio output devices, etc. The display subsystem may be a cathode ray tube (CRT), a flat-panel device, such as that using a liquid crystal display (LCD) or plasma display, a projection device, a touch screen, and the like. In general, use of the term "output device" is intended to include all possible types of devices and mechanisms for outputting information from computer system 1100 to a user or other computer. For example, user interface output devices may include, without limitation, a variety of display devices that visually convey text, graphics and audio/video information such as monitors, printers, speakers, headphones, automotive navigation systems, plotters, voice output devices, and modems.

[0122] Computer system 1100 may comprise a storage subsystem 1118 that comprises software elements, shown as being currently located within a system memory 1110. System

memory 1110 may store program instructions that are loadable and executable on processing unit 1104, as well as data generated during the execution of these programs.

[0123] Depending on the configuration and type of computer system 1100, system memory 1110 may be volatile (such as random access memory (RAM)) and/or non-volatile (such as read-only memory (ROM), flash memory, etc.) The RAM typically contains data and/or program modules that are immediately accessible to and/or presently being operated and executed by processing unit 1104. In some implementations, system memory 1110 may include multiple different types of memory, such as static random access memory (SRAM) or dynamic random access memory (DRAM). In some implementations, a basic input/output system (BIOS), containing the basic routines that help to transfer information between elements within computer system 1100, such as during start-up, may typically be stored in the ROM. By way of example, and not limitation, system memory 1110 also illustrates application programs 1112, which may include client applications, Web browsers, mid-tier applications, relational database management systems (RDBMS), etc., program data 1114, and an operating system 1116. By way of example, operating system 1116 may include various versions of Microsoft Windows®, Apple Macintosh®, and/or Linux operating systems, a variety of commercially-available UNIX® or UNIX-like operating systems (including without limitation the variety of GNU/Linux operating systems, the Google Chrome® OS, and the like) and/or mobile operating systems such as iOS, Windows® Phone, Android® OS, BlackBerry® 11 OS, and Palm® OS operating systems.

[0124] Storage subsystem 1118 may also provide a tangible computer-readable storage medium for storing the basic programming and data constructs that provide the functionality of some embodiments. Software (programs, code modules, instructions) that when executed by a processor provide the functionality described above may be stored in storage subsystem 1118. These software modules or instructions may be executed by processing unit 1104. Storage subsystem 1118 may also provide a repository for storing data used in accordance with the present invention.

[0125] Storage subsystem 1100 may also include a computer-readable storage media reader 1120 that can further be connected to computer-readable storage media 1122. Together and, optionally, in combination with system memory 1110, computer-readable storage media 1122 may comprehensively represent remote, local, fixed, and/or removable storage devices plus

storage media for temporarily and/or more permanently containing, storing, transmitting, and retrieving computer-readable information.

[0126] Computer-readable storage media 1122 containing code, or portions of code, can also include any appropriate media known or used in the art, including storage media and communication media, such as but not limited to, volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage and/or transmission of information. This can include tangible computer-readable storage media such as RAM, ROM, electronically erasable programmable ROM (EEPROM), flash memory or other memory technology, CD-ROM, digital versatile disk (DVD), or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or other tangible computer readable media. This can also include nontangible computer-readable media, such as data signals, data transmissions, or any other medium which can be used to transmit the desired information and which can be accessed by computing system 1100.

[0127] By way of example, computer-readable storage media 1122 may include a hard disk drive that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive that reads from or writes to a removable, nonvolatile magnetic disk, and an optical disk drive that reads from or writes to a removable, nonvolatile optical disk such as a CD ROM, DVD, and Blu-Ray® disk, or other optical media. Computer-readable storage media 1122 may include, but is not limited to, Zip® drives, flash memory cards, universal serial bus (USB) flash drives, secure digital (SD) cards, DVD disks, digital video tape, and the like. Computer-readable storage media 1122 may also include, solid-state drives (SSD) based on non-volatile memory such as flash-memory based SSDs, enterprise flash drives, solid state ROM, and the like, SSDs based on volatile memory such as solid state RAM, dynamic RAM, static RAM, DRAM-based SSDs, magnetoresistive RAM (MRAM) SSDs, and hybrid SSDs that use a combination of DRAM and flash memory based SSDs. The disk drives and their associated computer-readable media may provide non-volatile storage of computer-readable instructions, data structures, program modules, and other data for computer system 1100.

[0128] Communications subsystem 1124 provides an interface to other computer systems and networks. Communications subsystem 1124 serves as an interface for receiving data from and transmitting data to other systems from computer system 1100. For example, communications subsystem 1124 may enable computer system 1100 to connect to one or more devices via the

Internet. In some embodiments communications subsystem 1124 can include radio frequency (RF) transceiver components for accessing wireless voice and/or data networks (e.g., using cellular telephone technology, advanced data network technology, such as 3G, 4G or EDGE (enhanced data rates for global evolution), WiFi (IEEE 802.11 family standards, or other mobile communication technologies, or any combination thereof), global positioning system (GPS) receiver components, and/or other components. In some embodiments communications subsystem 1124 can provide wired network connectivity (e.g., Ethernet) in addition to or instead of a wireless interface.

[0129] In some embodiments, communications subsystem 1124 may also receive input communication in the form of structured and/or unstructured data feeds 1126, event streams 1128, event updates 1130, and the like on behalf of one or more users who may use computer system 1100.

[0130] By way of example, communications subsystem 1124 may be configured to receive data feeds 1126 in real-time from users of social networks and/or other communication services such as Twitter® feeds, Facebook® updates, web feeds such as Rich Site Summary (RSS) feeds, and/or real-time updates from one or more third party information sources.

[0131] Additionally, communications subsystem 1124 may also be configured to receive data in the form of continuous data streams, which may include event streams 1128 of real-time events and/or event updates 1130, which may be continuous or unbounded in nature with no explicit end. Examples of applications that generate continuous data may include, for example, sensor data applications, financial tickers, network performance measuring tools (e.g. network monitoring and traffic management applications), clickstream analysis tools, automobile traffic monitoring, and the like. Communications subsystem 1124 may also be configured to output the structured and/or unstructured data feeds 1126, event streams 1128, event updates 1130, and the like to one or more databases that may be in communication with one or more streaming data source computers coupled to computer system 1100.

[0132] Computer system 1100 can be one of various types, including a handheld portable device (e.g., an iPhone® cellular phone, an iPad® computing tablet, a PDA), a wearable device (e.g., a Google Glass® head mounted display), a PC, a workstation, a mainframe, a kiosk, a server rack, or any other data processing system.

[0133] Due to the ever-changing nature of computers and networks, the description of computer system 1100 depicted in the figure is intended only as a specific example. Many other configurations having more or fewer components than the system depicted in the figure are possible. For example, customized hardware might also be used and/or particular elements might be implemented in hardware, firmware, software (including applets), or a combination. Further, connection to other computing devices, such as network input/output devices, may be employed. Based on the disclosure and teachings provided herein, a person of ordinary skill in the art will appreciate other ways and/or methods to implement the various embodiments.

In the foregoing specification, aspects of the invention are described with reference to specific embodiments thereof, but those skilled in the art will recognize that the invention is not limited thereto. Various features and aspects of the above-described invention may be used individually or jointly. Further, embodiments can be utilized in any number of environments and applications beyond those described herein without departing from the broader spirit and scope of the specification. The specification and drawings are, accordingly, to be regarded as illustrative rather than restrictive.

WHAT IS CLAIMED IS:

1. A computer-implemented method comprising:

determining that a quantity of shards in a multi-shard system has changed;

in response to determining that the quantity of shards has changed, transitioning a client from a normal state, in which the client performs a particular type of operation relative to data items stored in the system in a first manner, to a rebalancing state, in which the client performs the particular type of operation relative to data items stored in the system in a second manner that differs from the first manner and without the client acquiring exclusive locks relative to any data items;

while the client is in the rebalancing state, determining, for one or more particular data items, destination shards that are separate from source shards on which the one or more particular data items were stored prior to the client's transition to the rebalancing state; and

moving the one or more data items from the source shards to the destination shards while the client is in the rebalancing state.

2. The computer-implemented method of Claim 1, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing, asynchronously to the moving of the one or more data items, and relative to a data item, an operation of the particular type in the second manner without acquiring an exclusive lock.

3. The computer-implemented method of Claim 1 or Claim 2, further comprising:

determining the destination shards for the one or more particular data items by hashing primary keys of the data items based at least in part on the quantity of shards in the multi-shard system following the shard quantity change; and

in response to determining that the one or more data items have been moved from the source shards to the destination shards, transitioning the client from the rebalancing state to the normal state.

4. The computer-implemented method of any of Claims 1-3, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing an add type of operation in the second manner relative to a first data item having a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which the first data item would have been added prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which the first data item is to be added after the shard quantity change;

determining that no data item on the source shard has the first data item's primary key;

determining that no data item on the destination shard has both a true tombstone attribute value and the first data item's primary key; and

inserting the first data item into the destination shard.

5. The computer-implemented method of any of Claims 1-4, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing an add type of operation in the second manner relative to a first data item having a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which the first data item would have been added prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which the first data item is to be added after the shard quantity change;

determining that no data item on the source shard has the first data item's primary key;

determining that a second data item on the destination shard has both a true tombstone attribute value and the first data item's primary key;

assigning, to attributes of the second data item, attribute values of the first data item; and

setting the second data item's tombstone attribute value to false.

6. The computer-implemented method of any of Claims 1-5, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing an update type of operation in the second manner relative to a first data item having a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which the first data item would have been located prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which the first data item is to be located after the shard quantity change;

determining that no data item on the source shard has the first data item's primary key;

determining that a second data item on the destination shard has both a false tombstone attribute value and the first data item's primary key; and

assigning, to attributes of the second data item, attribute values of the first data item.

7. The computer-implemented method of any of Claims 1-6, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing an update type of operation in the second manner relative to a first data item having a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which the first data item would have been located prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which the first data item is to be located after the shard quantity change;

determining that a second data item on the source shard has both the first data item's primary key and a false tombstone attribute value;

determining that a third data item on the destination shard has the first data item's primary key;

assigning, to attributes of the third data item, attribute values of the first data item; and

setting a version number of the third data item to a value different from a version number of the second data item.

8. The computer-implemented method of any of Claims 1-7, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing an update type of operation in the second manner relative to a first data item having a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which the first data item would have been located prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which the first data item is to be located after the shard quantity change;

determining that a second data item on the source shard has both the first data item's primary key and a false tombstone attribute value;

determining no data item on the destination shard has the first data item's primary key; and

inserting the first data item into the destination shard with a version number that differs from a version number of the second data item.

9. The computer-implemented method of any of Claims 1-8, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing a delete type of operation in the second manner relative to a first data item having a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which the first data item would have been located prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which the first data item is to be located after the shard quantity change;

determining that a second data item on the source shard has the first data item's primary key;

upserting the first data item into the destination shard with a version number that differs from a version number of the second data item; and

setting a tombstone attribute value of the first data item on the destination shard to true.

10. The computer-implemented method of any of Claims 1-9, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing a delete type of operation in the second manner relative to a first data item having a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which the first data item would have been located prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which the first data item is to be located after the shard quantity change;

determining that no data item on the source shard has the first data item's primary key;

determining that a second data item on the destination shard has the first data item's primary key; and

setting a tombstone attribute value of the second data item to true.

11. The computer-implemented method of any of Claims 1-10, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing a get type of operation in the second manner relative to a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which a data item having the primary key would have been located prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which a data item having the primary key is to be located after the shard quantity change;

determining that a first data item on the destination shard has the primary key and a false tombstone attribute value; and

reading attribute values of the first data item.

12. The computer-implemented method of any of Claims 1-11, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing a get type of operation in the second manner relative to a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which a data item having the primary key would have been located prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which a data item having the primary key is to be located after the shard quantity change;

determining that no data item on the destination shard has the primary key;

determining that a first data item on the source shard has the primary key and a false tombstone attribute value; and

reading attribute values of the first data item.

13. The computer-implemented method of any of Claims 1-12, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client performing a get type of operation in the second manner relative to a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which a data item having the primary key would have been located prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which a data item having the primary key is to be located after the shard quantity change;

determining, at a first moment in time, that no data item on the destination shard has the primary key;

determining that no data item on the source shard has the primary key;

determining, at a second moment in time, that a first data item on the destination shard has the primary key and a false tombstone attribute value; and

reading attribute values of the first data item.

14. The computer-implemented method of any of Claims 1-13, wherein moving the one or more data items from the source shards to the destination shards comprises:

incrementing a system-wide version number; and

for each first data item having a version number that is less than the system-wide version number, determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a first destination shard on which the first data item is to be located after the shard quantity change;

for each second data item that (a) has a version number that is less than the system-wide version number and (b) is located on a source shard that differs from a second destination shard on which the second data item is to be located after the shard quantity change, inserting, into the second destination shard, a copy of the second data item having the system-wide version number;

for each third data item that (a) has a version number that is less than the system-wide version number and (b) is already located on a third destination shard on which the third data item is to be located after the shard quantity change, changing a version number of the third data item to the system-wide version number.

15. The computer-implemented method of any of Claims 1-14, wherein moving the one or more data items from the source shards to the destination shards comprises incrementing a system-wide version number, and further comprising:

determining a set of queries that are pending against one or more shards in the multi-shard system as of a time that the movement of the one or more data items completes;

waiting for all queries in the set of queries to finish;

after all queries in the set of queries have finished, removing, from all shards in the multi-shard system, all data item copies having version number attribute values that differ from the system-wide version number; and

after all queries in the set of queries have finished, removing, from all shards in the multi-shard system, all data item copies having true tombstone attribute values.

16. The computer-implemented method of any of Claims 1-15, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, and for each particular shard in the multi-shard system, populating a preliminary result queue for that particular shard with data items that both (a) are located on that particular shard and (b) satisfy specified query criteria;

for each particular shard's preliminary result queue, sorting data items in that shard's preliminary result queue based at least in part on primary keys of the data item in that shard's preliminary result queue;

until every shard's preliminary result queue is empty, repeatedly performing operations comprising:

selecting a first data item from a set comprising data items currently located at tops of preliminary result queues of all of the shards in the multi-shard system,

adding, to a final result set, each first data item that does not have a true tombstone attribute value, and

removing, from every shard's preliminary result queue, all data item copies having a primary key that matches a primary key of the first data item; and

after every shard's preliminary result queue is empty, returning data items in the final result set as query result.

17. The computer-implemented method of Claim 16, wherein selecting the first data item from the set comprising data items currently located at tops of preliminary result queues of all of the shards in the multi-shard system comprises:

selecting, from the set, a subset of data items having a smallest primary key of primary keys of data items in the set; and

selecting, as the first data item, a data item having a largest version number attribute value of version number attribute values of data item in the subset.

18. The computer-implemented method of any of Claims 1-17, further comprising:

while the one or more data items are being moved from the source shards to the destination shards, the client attempting and failing to perform a delete type of operation in the second manner relative to a first data item having a primary key at least in part by:

determining, based at least in part on a quantity of shards in the multi-shard system preceding the shard quantity change, a particular source shard on which the first data item would have been located prior to the shard quantity change;

determining, based at least in part on a quantity of shards in the multi-shard system following the shard quantity change, a particular destination shard on which the first data item is to be located after the shard quantity change;

determining that no data item on the source shard has the first data item's primary key;

determining that no data item on the destination shard has the first data item's primary key; and

generating data that indicates that the delete operation failed due to the first data item not existing.

19. A system comprising:

a plurality of database shards that store data items;

a plurality of clients that are configured to transition, in response to a change in a quantity of the plurality of database shards, from a normal state, in which each client of the plurality of clients is configured to perform a particular type of operation relative to the data

items in a first manner, to a rebalancing state, in which each client of the plurality of clients is configured to perform the particular type of operation relative to the data items in a second manner that differs from the first manner and without the client acquiring exclusive locks relative to any data items; and

at least one computing device configured to (a) determine, while the plurality of clients are in the rebalancing state, and for one or more particular data items, destination shards that are separate from source shards on which the one or more particular data items were stored while each client of the plurality of clients was in the normal state, and (b) move the one or more data items from the source shards to the destination shards while each client of the plurality of clients is in the rebalancing state.

20. A computer-readable storage memory storing processor-executable instructions comprising:

instructions to cause one or more processors to determine that a quantity of shards in a multi-shard system has changed;

instructions to cause one or more processors to transition a client, in response to a determination that the quantity of shards has changed, from a normal state, in which the client performs a particular type of operation relative to data items stored in the system in a first manner, to a rebalancing state, in which the client performs the particular type of operation relative to data items stored in the system in a second manner that differs from the first manner and without the client acquiring exclusive locks relative to any data items;

instructions to cause one or more processors to determine, while the client is in the rebalancing state, and for one or more particular data items, destination shards that are separate from source shards on which the one or more particular data items were stored prior to the client's transition to the rebalancing state; and

instructions to cause one or more processors to move the one or more data items from the source shards to the destination shards while the client is in the rebalancing state.

21. A system comprising means for performing operations as recited in any of Claims 1-18.

22. A computer program product storing processor-executable instructions for performing operations as recited in any of Claims 1-18.

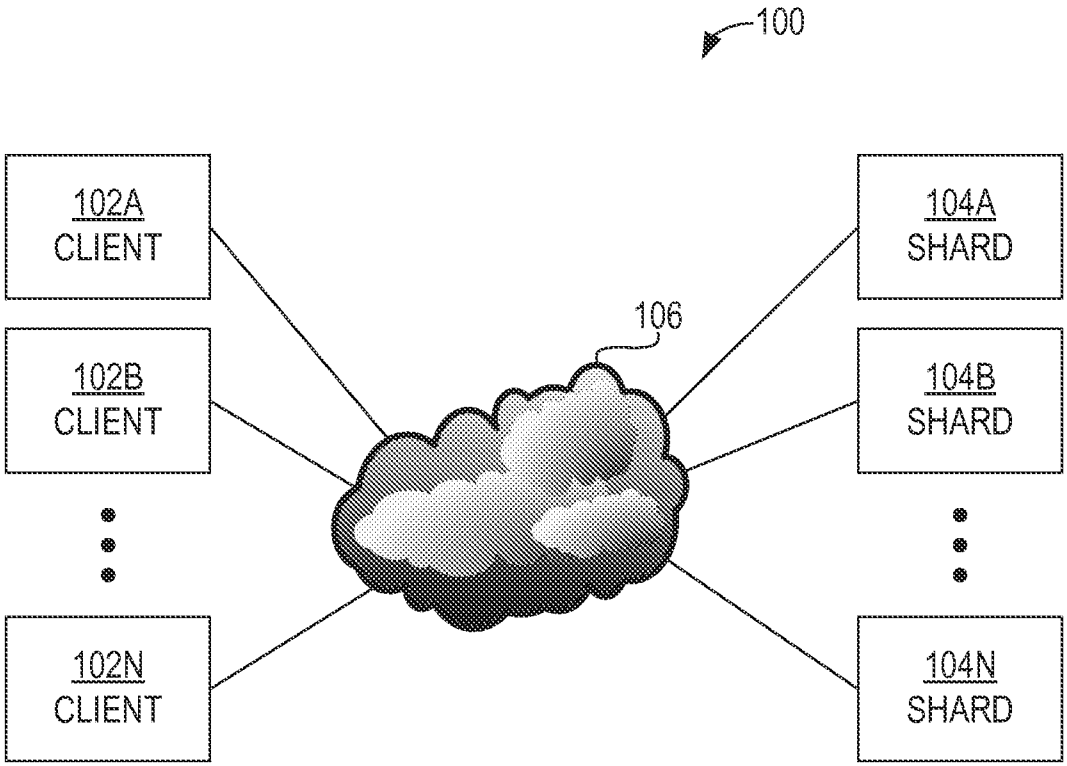
100
102A
CLIENT
102B
CLIENT
102N
CLIENT
106
104A
SHARD
104B
SHARD
104N
SHARD

FIG. 1

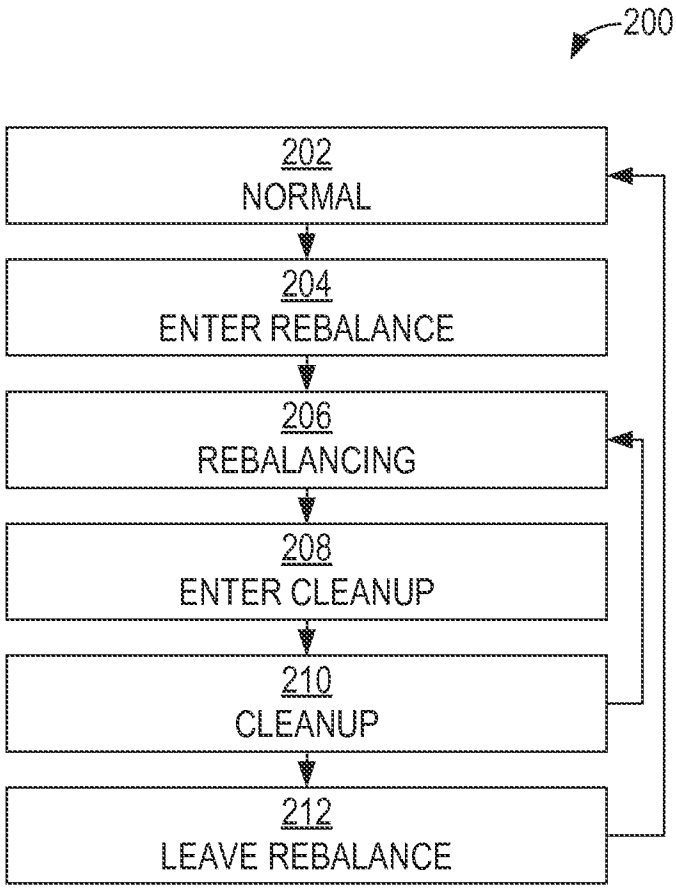
200
202
NORMAL
204
ENTER REBALANCE
206
REBALANCING
208
ENTER CLEANUP
210
CLEANUP
212
LEAVE REBALANCE

FIG. 2

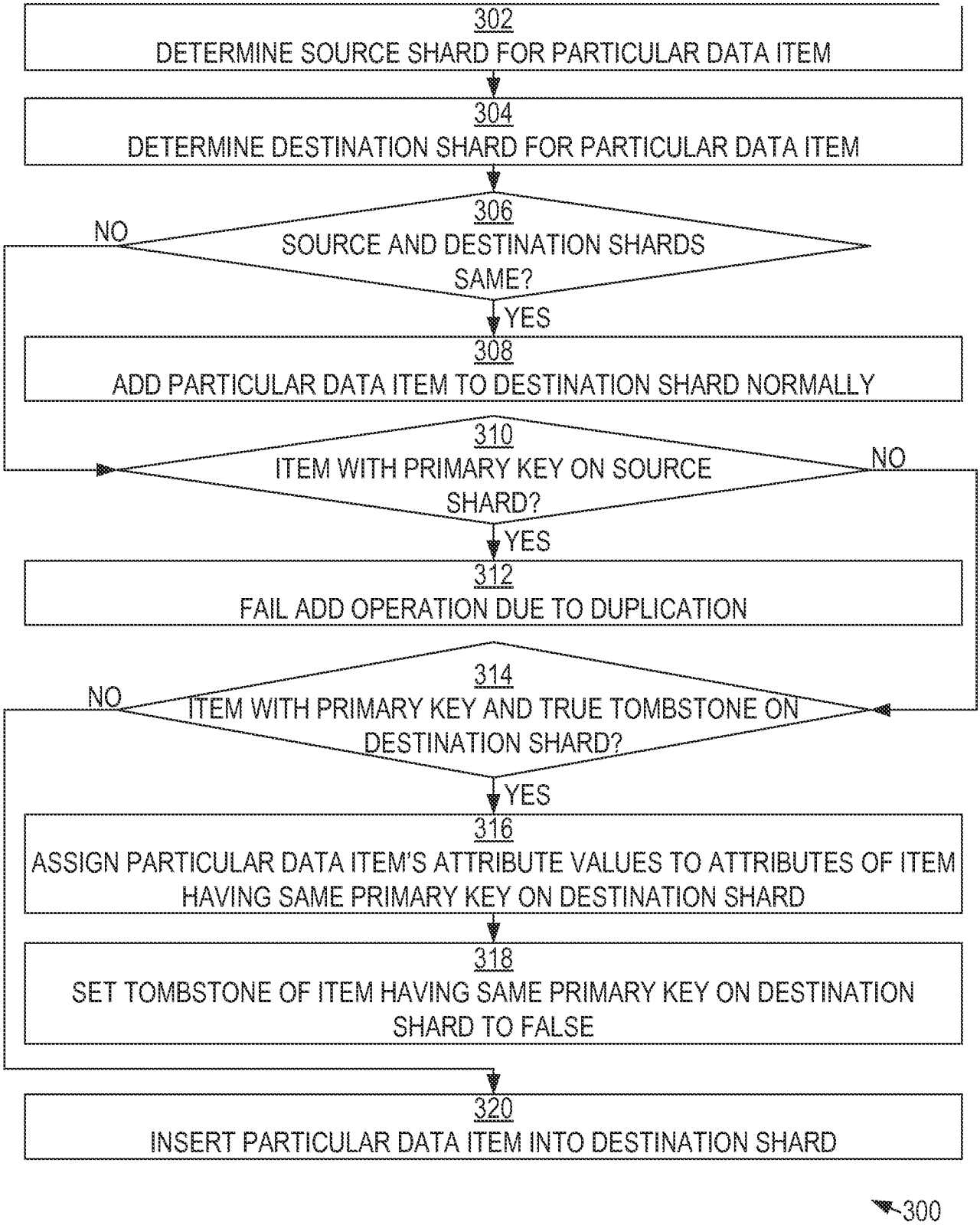
302
DETERMINE SOURCE SHARD FOR PARTICULAR DATA ITEM
304
DETERMINE DESTINATION SHARD FOR PARTICULAR DATA ITEM
306
SOURCE AND DESTINATION SHARDS SAME?
NO
YES
308
ADD PARTICULAR DATA ITEM TO DESTINATION SHARD NORMALLY
310
ITEM WITH PRIMARY KEY ON SOURCE SHARD?
NO
YES
312
FAIL ADD OPERATION DUE TO DUPLICATION
314
ITEM WITH PRIMARY KEY AND TRUE TOMBSTONE ON DESTINATION SHARD?
NO
YES
316
ASSIGN PARTICULAR DATA ITEM'S ATTRIBUTE VALUES TO ATTRIBUTES OF ITEM HAVING SAME PRIMARY KEY ON DESTINATION SHARD
318
SET TOMBSTONE OF ITEM HAVING SAME PRIMARY KEY ON DESTINATION SHARD TO FALSE
320
INSERT PARTICULAR DATA ITEM INTO DESTINATION SHARD
300

FIG. 3

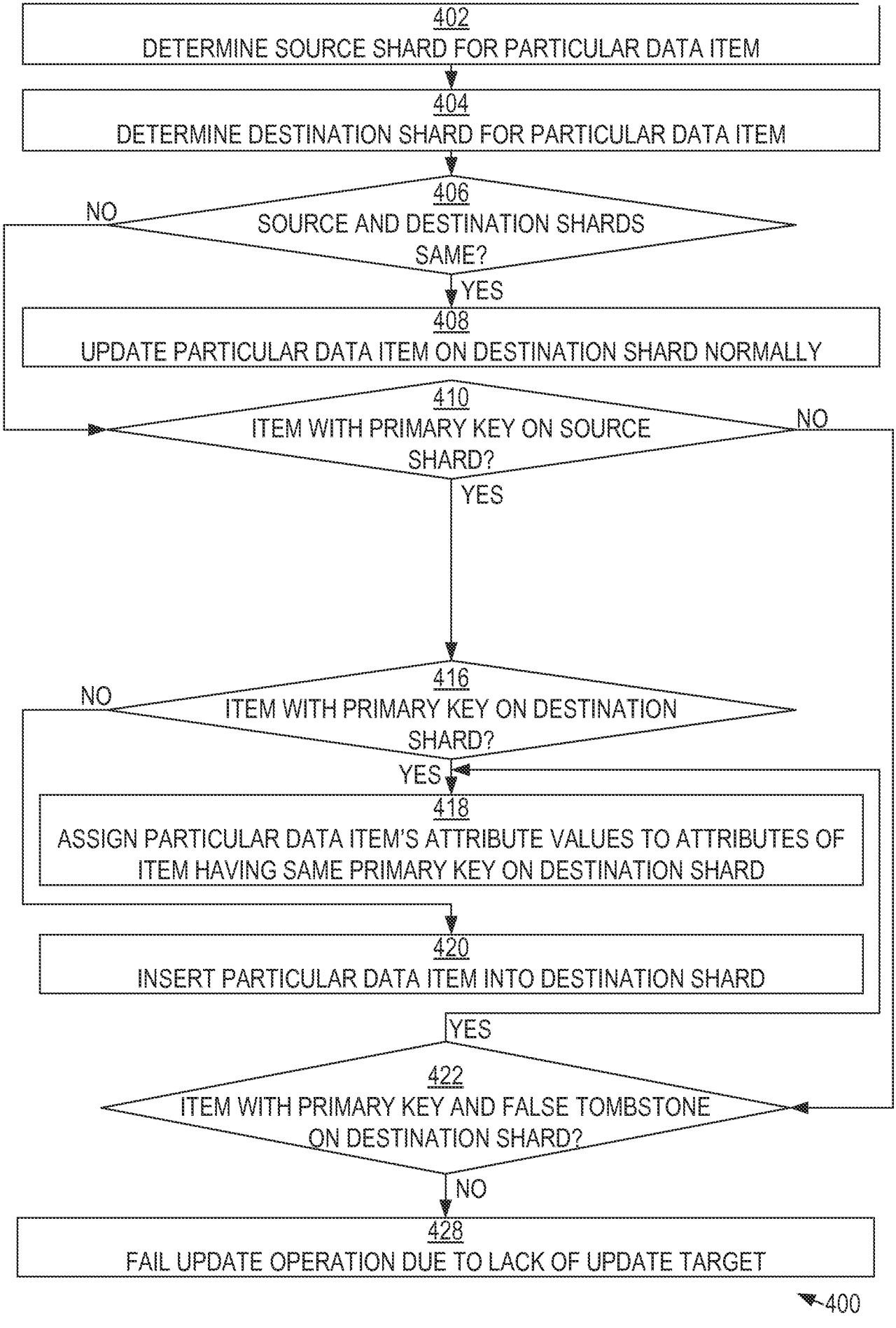
402
DETERMINE SOURCE SHARD FOR PARTICULAR DATA ITEM
404
DETERMINE DESTINATION SHARD FOR PARTICULAR DATA ITEM
406
SOURCE AND DESTINATION SHARDS SAME?
NO
YES
408
UPDATE PARTICULAR DATA ITEM ON DESTINATION SHARD NORMALLY
410
ITEM WITH PRIMARY KEY ON SOURCE SHARD?
NO
YES
416
ITEM WITH PRIMARY KEY ON DESTINATION SHARD?
NO
YES
418
ASSIGN PARTICULAR DATA ITEM'S ATTRIBUTE VALUES TO ATTRIBUTES OF ITEM HAVING SAME PRIMARY KEY ON DESTINATION SHARD
420
INSERT PARTICULAR DATA ITEM INTO DESTINATION SHARD
YES
422
ITEM WITH PRIMARY KEY AND FALSE TOMBSTONE ON DESTINATION SHARD?
NO
428
FAIL UPDATE OPERATION DUE TO LACK OF UPDATE TARGET
400

FIG. 4

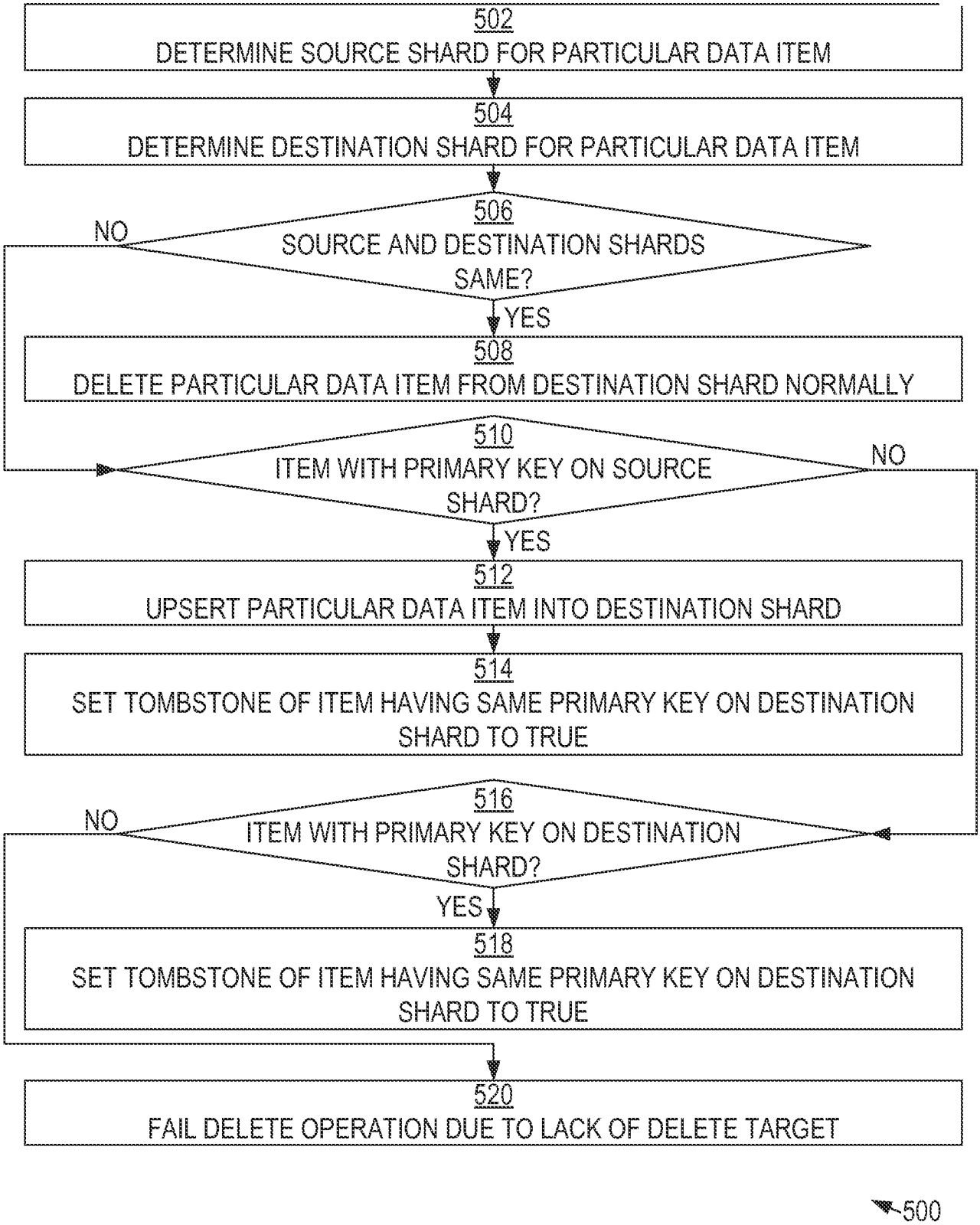
502
DETERMINE SOURCE SHARD FOR PARTICULAR DATA ITEM
504
DETERMINE DESTINATION SHARD FOR PARTICULAR DATA ITEM
506
SOURCE AND DESTINATION SHARDS SAME?
NO
YES
508
DELETE PARTICULAR DATA ITEM FROM DESTINATION SHARD NORMALLY
510
ITEM WITH PRIMARY KEY ON SOURCE SHARD?
NO
YES
512
UPSERT PARTICULAR DATA ITEM INTO DESTINATION SHARD
514
SET TOMBSTONE OF ITEM HAVING SAME PRIMARY KEY ON DESTINATION SHARD TO TRUE
516
ITEM WITH PRIMARY KEY ON DESTINATION SHARD?
NO
YES
518
SET TOMBSTONE OF ITEM HAVING SAME PRIMARY KEY ON DESTINATION SHARD TO TRUE
520
FAIL DELETE OPERATION DUE TO LACK OF DELETE TARGET
500

FIG. 5

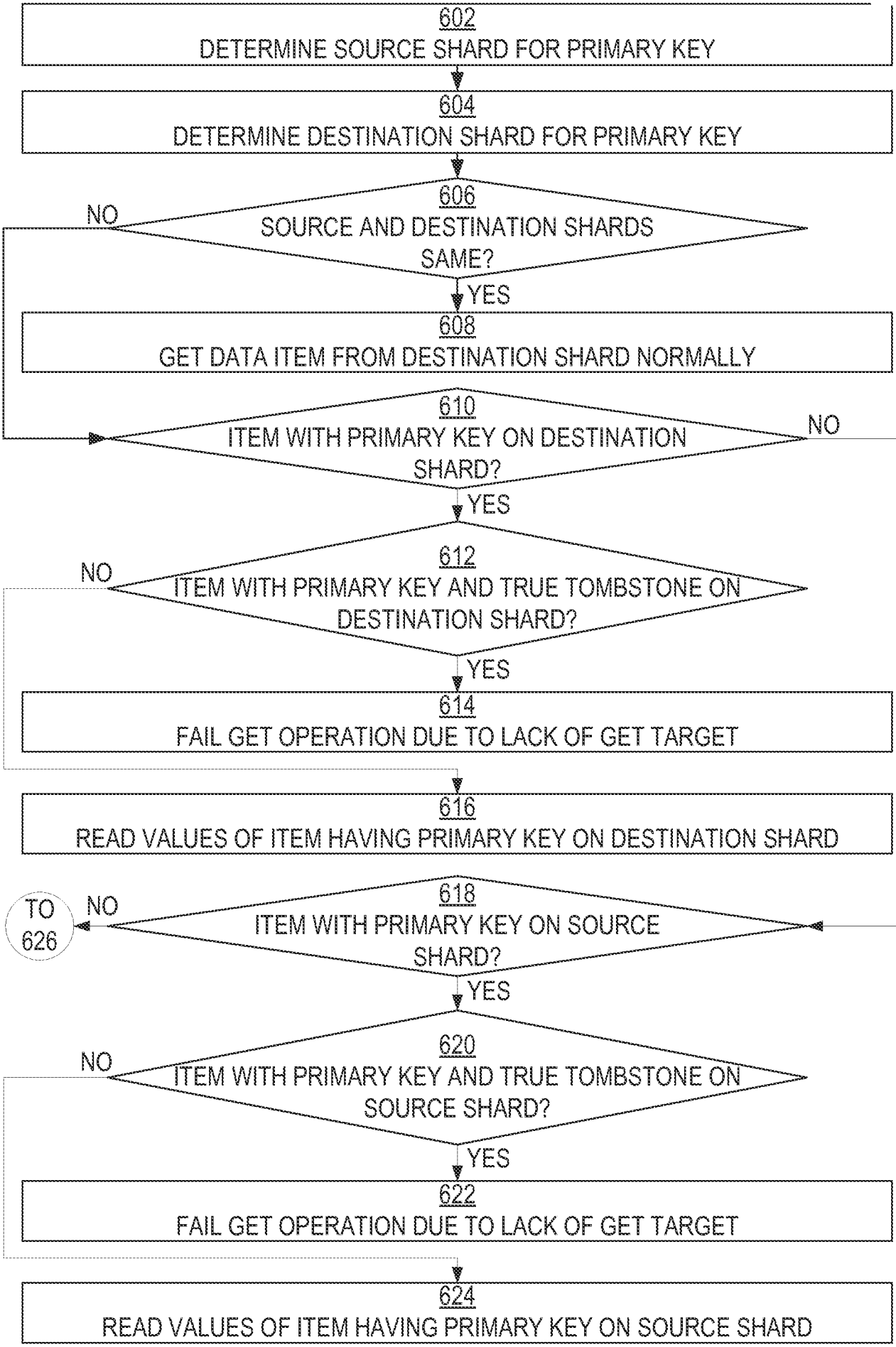
602
DETERMINE SOURCE SHARD FOR PRIMARY KEY
604
DETERMINE DESTINATION SHARD FOR PRIMARY KEY
606
SOURCE AND DESTINATION SHARDS SAME?
NO
YES
608
GET DATA ITEM FROM DESTINATION SHARD NORMALLY
610
ITEM WITH PRIMARY KEY ON DESTINATION SHARD?
NO
YES
612
ITEM WITH PRIMARY KEY AND TRUE TOMBSTONE ON DESTINATION SHARD?
NO
YES
614
FAIL GET OPERATION DUE TO LACK OF GET TARGET
616
READ VALUES OF ITEM HAVING PRIMARY KEY ON DESTINATION SHARD
TO 626
618
ITEM WITH PRIMARY KEY ON SOURCE SHARD?
NO
YES
620
ITEM WITH PRIMARY KEY AND TRUE TOMBSTONE ON SOURCE SHARD?
NO
YES
622
FAIL GET OPERATION DUE TO LACK OF GET TARGET
624
READ VALUES OF ITEM HAVING PRIMARY KEY ON SOURCE SHARD
600

FIG. 6A

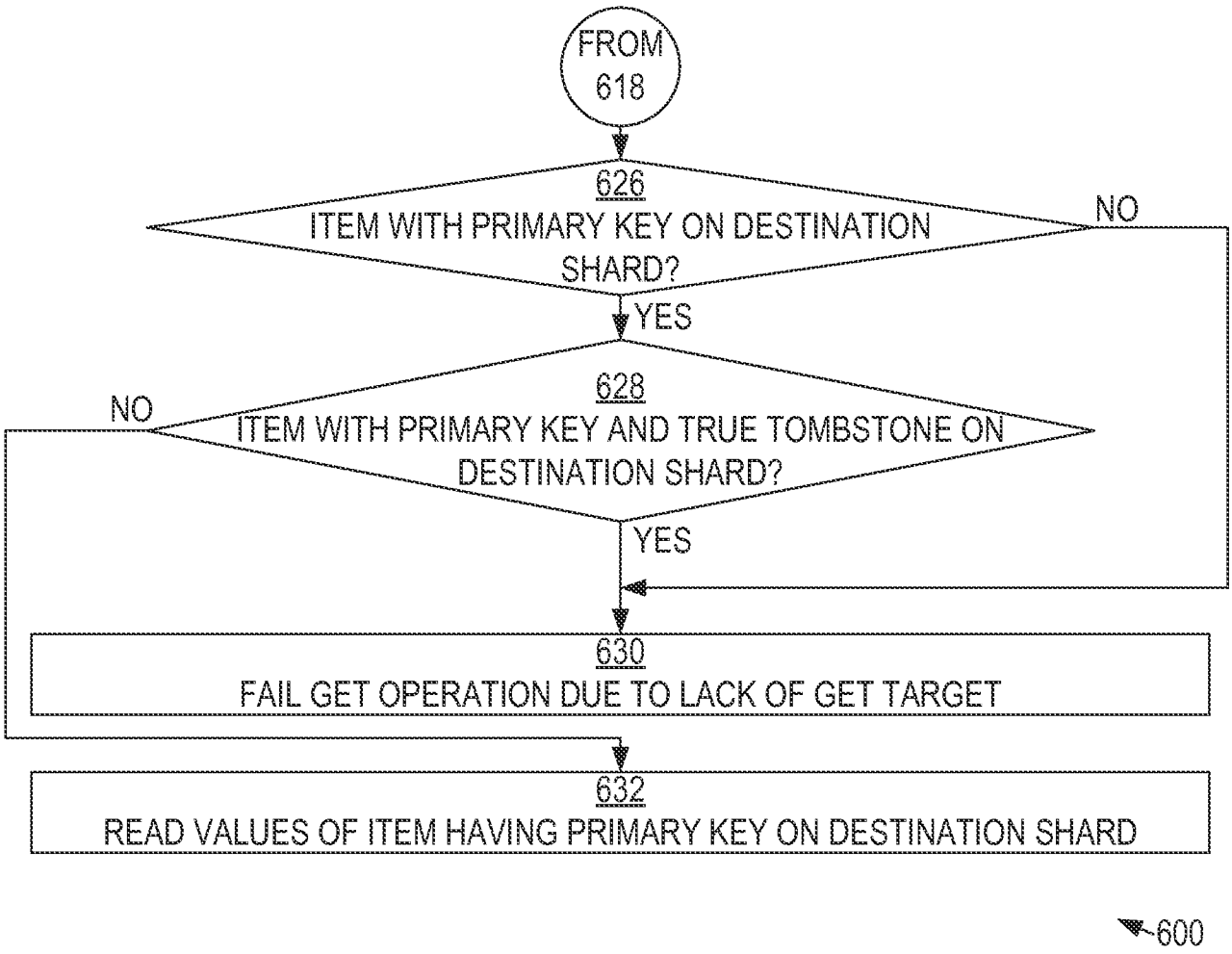
FROM 618
626
ITEM WITH PRIMARY KEY ON DESTINATION SHARD?
NO
YES
628
ITEM WITH PRIMARY KEY AND TRUE TOMBSTONE ON DESTINATION SHARD?
NO
YES
630
FAIL GET OPERATION DUE TO LACK OF GET TARGET
632
READ VALUES OF ITEM HAVING PRIMARY KEY ON DESTINATION SHARD
600

FIG. 6B

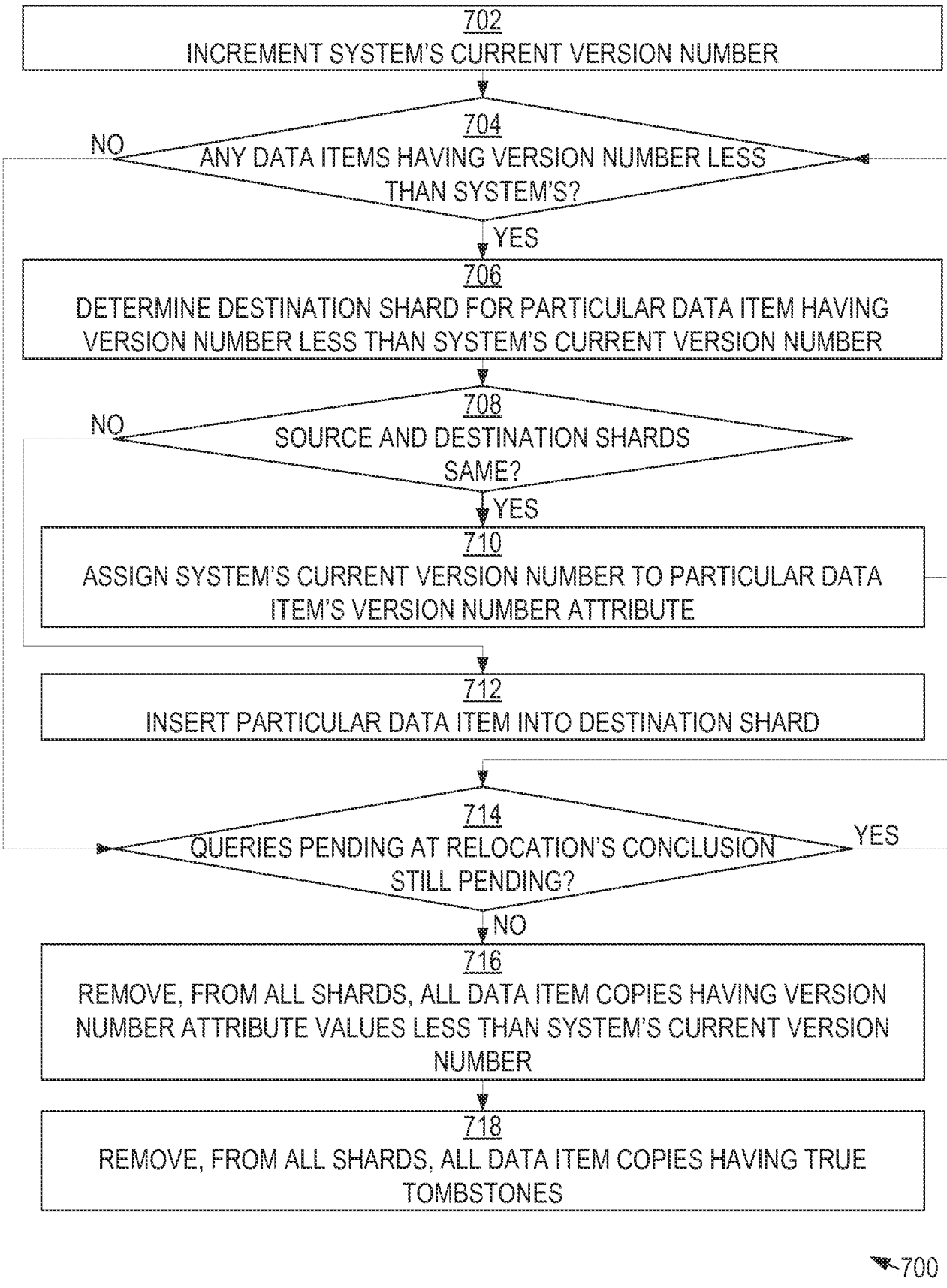
702
INCREMENT SYSTEM'S CURRENT VERSION NUMBER
704
ANY DATA ITEMS HAVING VERSION NUMBER LESS THAN SYSTEM'S?
NO
YES
706
DETERMINE DESTINATION SHARD FOR PARTICULAR DATA ITEM HAVING VERSION NUMBER LESS THAN SYSTEM'S CURRENT VERSION NUMBER
708
SOURCE AND DESTINATION SHARDS SAME?
NO
YES
710
ASSIGN SYSTEM'S CURRENT VERSION NUMBER TO PARTICULAR DATA ITEM'S VERSION NUMBER ATTRIBUTE
712
INSERT PARTICULAR DATA ITEM INTO DESTINATION SHARD
714
QUERIES PENDING AT RELOCATION'S CONCLUSION STILL PENDING?
YES
NO
716
REMOVE, FROM ALL SHARDS, ALL DATA ITEM COPIES HAVING VERSION NUMBER ATTRIBUTE VALUES LESS THAN SYSTEM'S CURRENT VERSION NUMBER
718
REMOVE, FROM ALL SHARDS, ALL DATA ITEM COPIES HAVING TRUE TOMBSTONES
700

FIG. 7

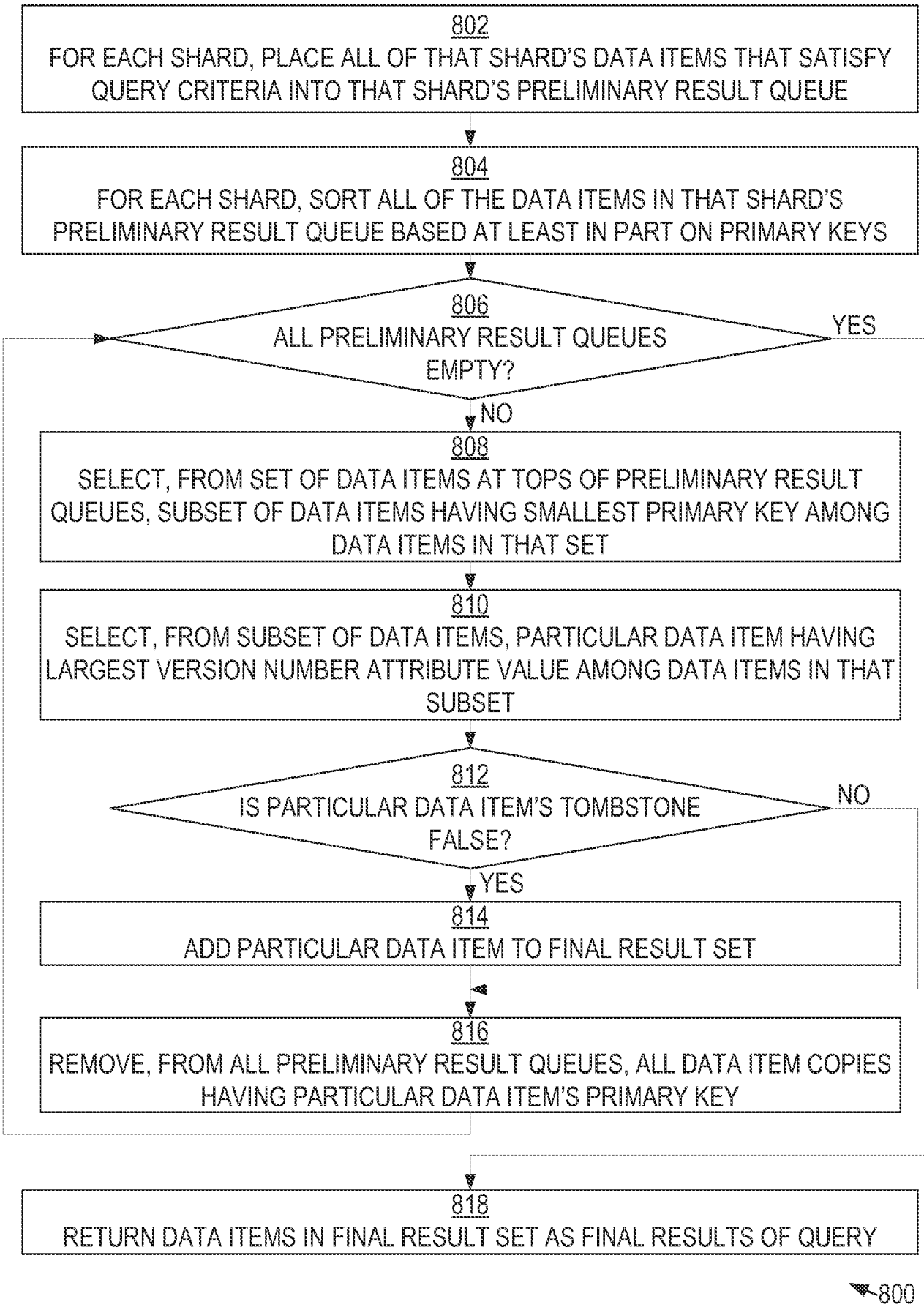
802
FOR EACH SHARD, PLACE ALL OF THAT SHARD'S DATA ITEMS THAT SATISFY QUERY CRITERIA INTO THAT SHARD'S PRELIMINARY RESULT QUEUE
804
FOR EACH SHARD, SORT ALL OF THE DATA ITEMS IN THAT SHARD'S PRELIMINARY RESULT QUEUE BASED AT LEAST IN PART ON PRIMARY KEYS
806
ALL PRELIMINARY RESULT QUEUES EMPTY?
YES
NO
808
SELECT, FROM SET OF DATA ITEMS AT TOPS OF PRELIMINARY RESULT QUEUES, SUBSET OF DATA ITEMS HAVING SMALLEST PRIMARY KEY AMONG DATA ITEMS IN THAT SET
810
SELECT, FROM SUBSET OF DATA ITEMS, PARTICULAR DATA ITEM HAVING LARGEST VERSION NUMBER ATTRIBUTE VALUE AMONG DATA ITEMS IN THAT SUBSET
812
IS PARTICULAR DATA ITEM'S TOMBSTONE FALSE?
NO
YES
814
ADD PARTICULAR DATA ITEM TO FINAL RESULT SET
816
REMOVE, FROM ALL PRELIMINARY RESULT QUEUES, ALL DATA ITEM COPIES HAVING PARTICULAR DATA ITEM'S PRIMARY KEY
818
RETURN DATA ITEMS IN FINAL RESULT SET AS FINAL RESULTS OF QUERY
800

FIG. 8

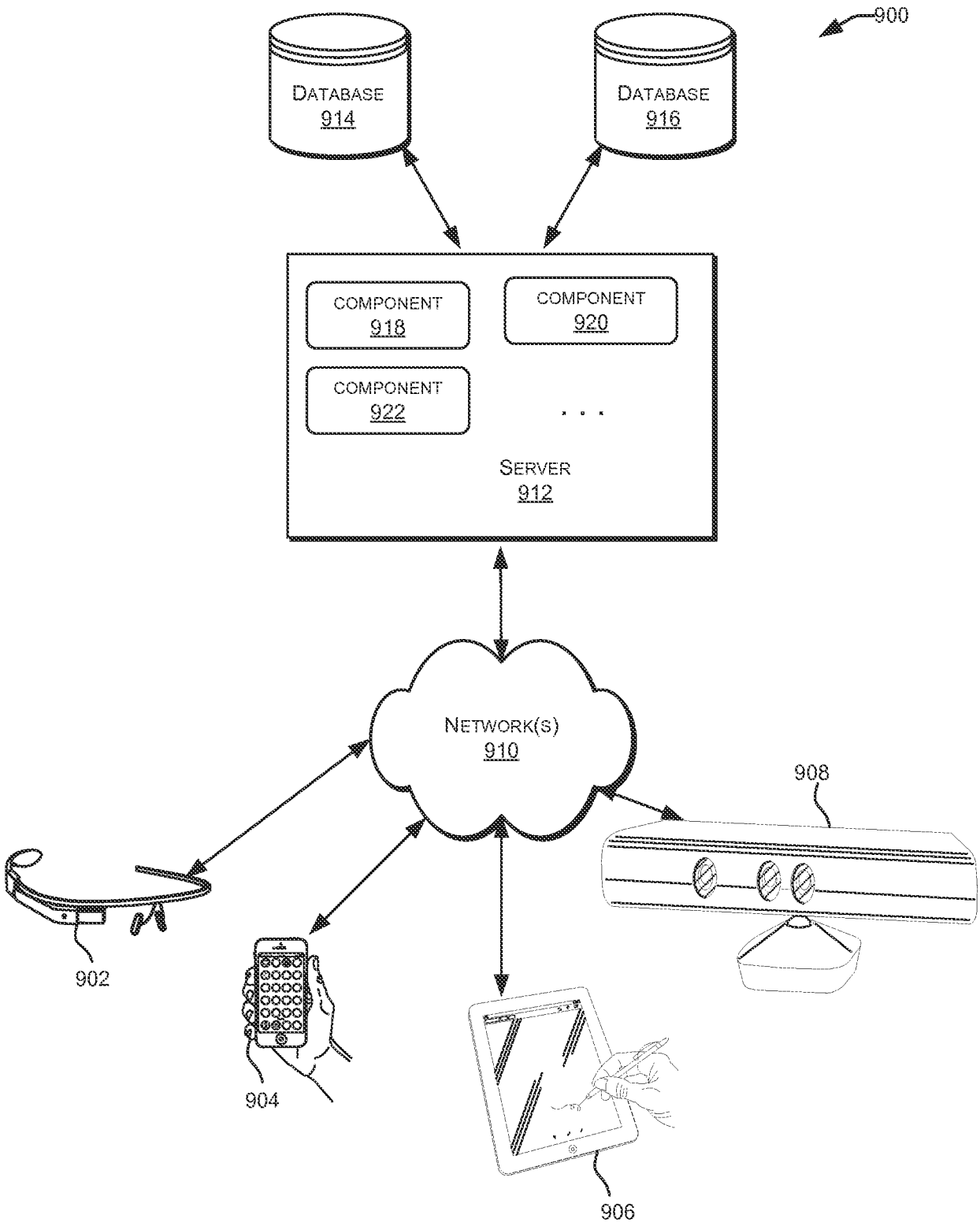
900
DATABASE
914
DATABASE
916
COMPONENT
918
COMPONENT
920
COMPONENT
922
. . .
SERVER
912
NETWORK(S)
910
908
902
904
906

FIG. 9

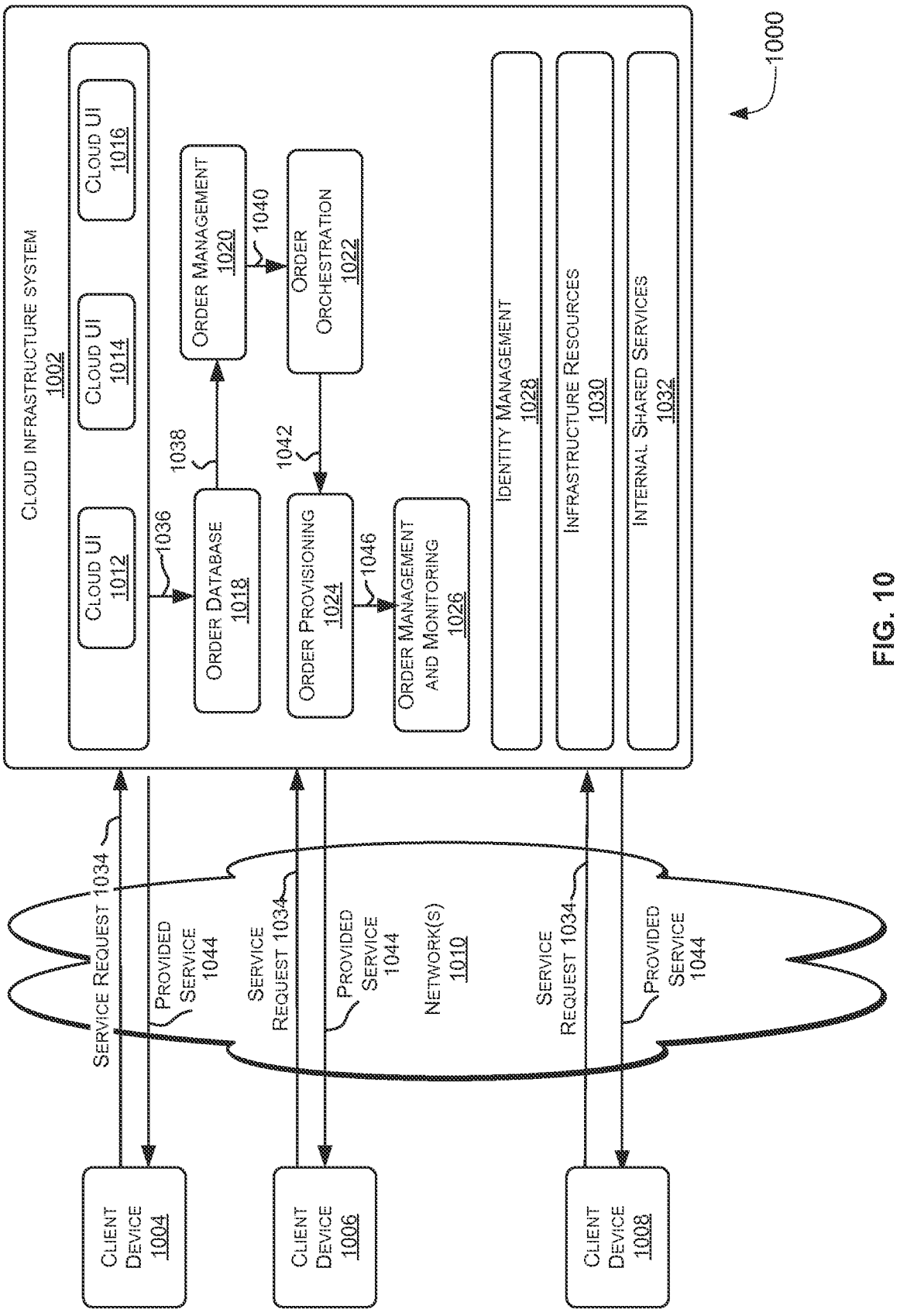
1000
CLOUD INFRASTRUCTURE SYSTEM
1002
CLOUD UI
1012
CLOUD UI
1014
CLOUD UI
1016
ORDER DATABASE
1018
ORDER MANAGEMENT
1020
ORDER
ORCHESTRATION
1022
ORDER PROVISIONING
1024
ORDER MANAGEMENT
AND MONITORING
1026
IDENTITY MANAGEMENT
1028
INFRASTRUCTURE RESOURCES
1030
INTERNAL SHARED SERVICES
1032
1036
1038
1040
1042
1046
SERVICE REQUEST 1034
PROVIDED
SERVICE
1044
SERVICE
REQUEST 1034
PROVIDED
SERVICE
1044
NETWORK(S)
1010
SERVICE
REQUEST 1034
PROVIDED
SERVICE
1044
CLIENT
DEVICE
1004
CLIENT
DEVICE
1006
CLIENT
DEVICE
1008

FIG. 10

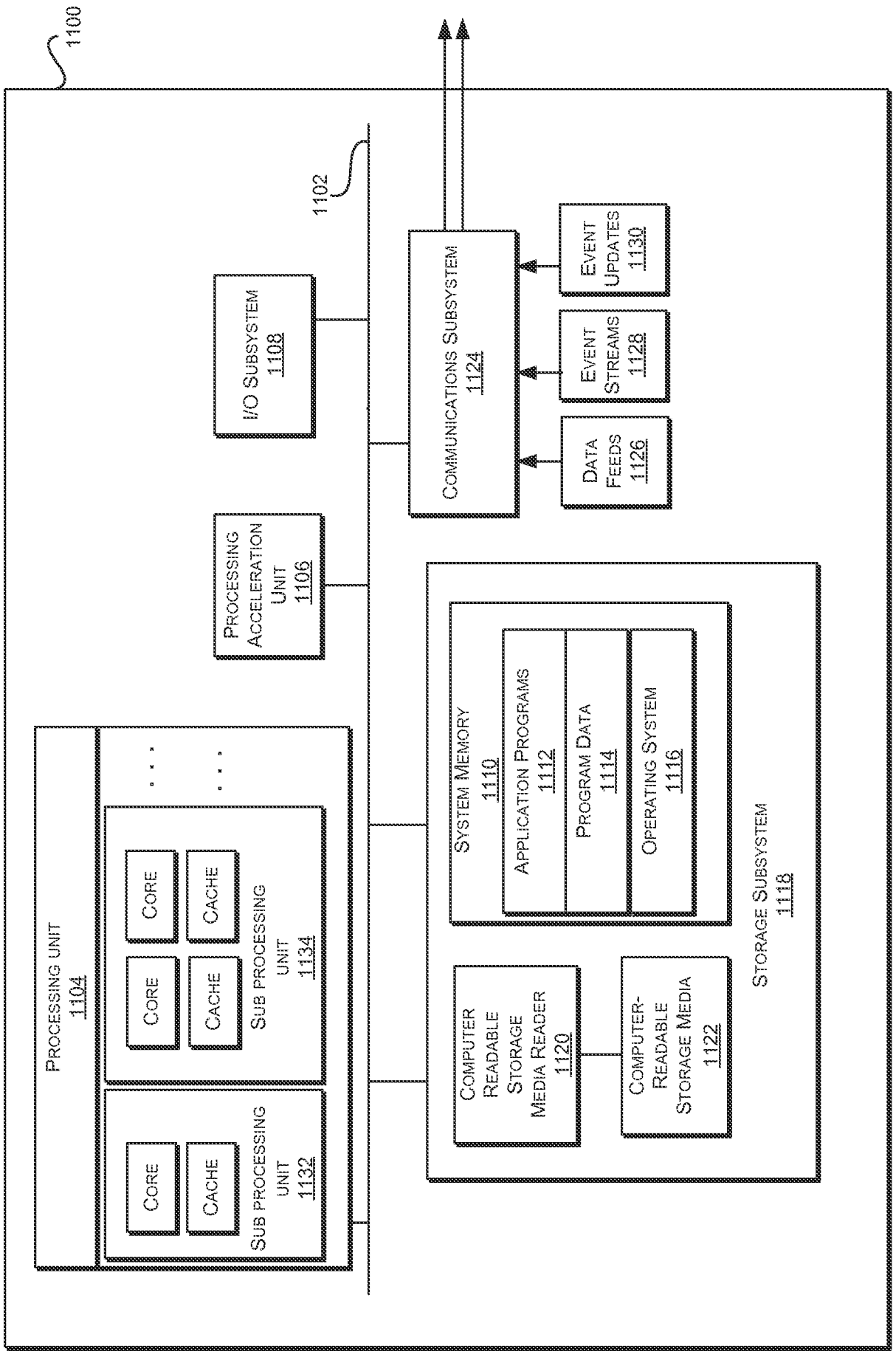
1100
1102
PROCESSING UNIT 1104
CORE
CACHE
SUB PROCESSING UNIT 1132
CORE
CACHE
SUB PROCESSING UNIT 1134
PROCESSING ACCELERATION UNIT 1106
I/O SUBSYSTEM 1108
COMMUNICATIONS SUBSYSTEM 1124
DATA FEEDS 1126
EVENT STREAMS 1128
EVENT UPDATES 1130
STORAGE SUBSYSTEM 1118
SYSTEM MEMORY 1110
APPLICATION PROGRAMS 1112
PROGRAM DATA 1114
OPERATING SYSTEM 1116
COMPUTER READABLE STORAGE MEDIA READER 1120
COMPUTER-READABLE STORAGE MEDIA 1122

FIG. 11

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/043599

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/30
ADD.

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

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)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2011/282832 A1 (RISHEL WILLIAM S [US] ET AL) 17 November 2011 (2011-11-17) abstract; figure 1 paragraphs [0002] - [0008], [0015], [0041] - [0045], [0050] - [0063] ----- -/--	1-22

[X] Further documents are listed in the continuation of Box C.

[X] 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 application or patent 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
1 October 2014	14/10/2014

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Hackelbusch, Richard

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/043599

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Baron Schwartz ET AL: "High Performance MySQL: Optimization, Backups, Replication, and More", , June 2008 (2008-06), pages 428-429, XP055138428, ISBN: 978-0-59-610171-8 Retrieved from the Internet: URL:http://books.google.de/books?id=BL0NNoFPuAQC&printsec=frontcover&hl=de#v=onepage&q&f=false [retrieved on 2014-09-05] section "Rebalancing shards"; page 428 - page 429 -----	1,2, 19-22

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2014/043599

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011282832 A1	17-11-2011	NONE	