Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2746416 C 2015/10/20

(11)(21) 2 746 416

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

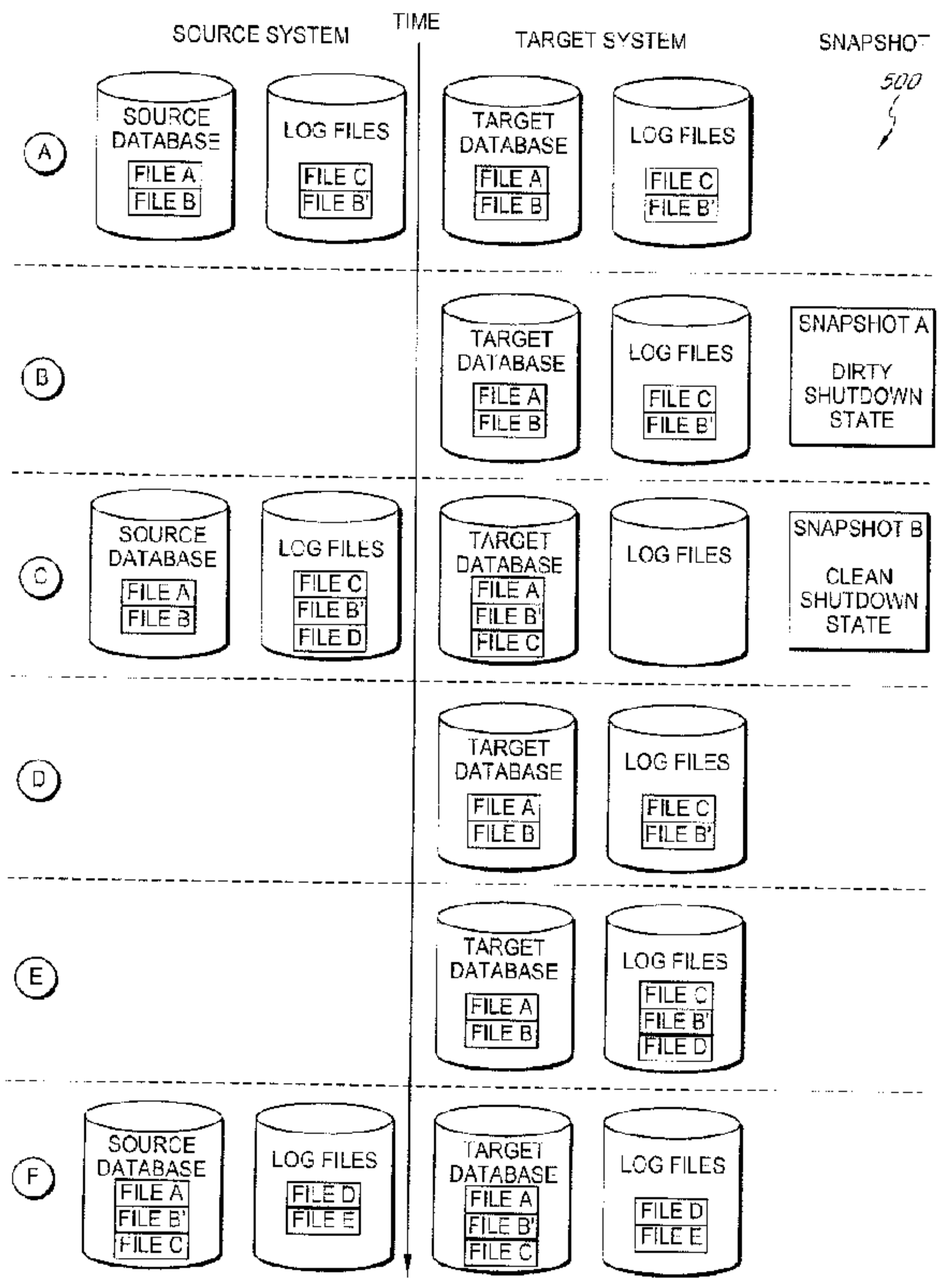
(86) Date de dépôt PCT/PCT Filing Date: 2009/12/04
(87) Date publication PCT/PCT Publication Date: 2010/06/17
(45) Date de délivrance/Issue Date: 2015/10/20
(85) Entrée phase nationale/National Entry: 2011/06/09
(86) N° demande PCT/PCT Application No.: US 2009/066880
(87) N° publication PCT/PCT Publication No.: 2010/068570
(30) Priorités/Priorities: 2008/12/10 (US61/121,418);
2009/04/15 (US12/424,115)

(51) Cl.Int./Int.Cl. *G06F 11/14* (2006.01)
(72) Inventeur/Inventor:
NGO, DAVID, US

(73) Propriétaire/Owner:
COMMVAULT SYSTEMS, INC., US

(74) Agent: BORDEN LADNER GERVAIS LLP

(54) Titre : PROCEDE ET SYSTEME DE GESTION DE DONNEES DE BASE DUPLIQUEES
(54) Title: METHOD AND SYSTEM FOR MANAGING REPLICATED DATABASE DATA



(57) Abrégé/Abstract:
Systems and methods for replicating database data and generating read-only copies of the replicated data in a clean shutdown state. For example, systems can include a tracking module (e.g., a filter driver) that monitors transactions from a database application to a source storage device to generate log entries having at least one marker indicating a known good state of the application. The systems further include a computer coupled to a target storage device comprising a database and log files. The computer processes the transactions, based on the log entries, to replicate data to the target storage device; performs a first snapshot on data stored in the database and log files; replays into the database data stored in the log files; performs another snapshot on the database; and reverts the database back to a state in which the database existed at the time of the first snapshot.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 June 2010 (17.06.2010)

PCT

(10) International Publication Number
WO 2010/068570 A1

(51) **International Patent Classification**:
G06F 11/14 (2006.01)

(21) **International Application Number**:
PCT/US2009/066880

(22) **International Filing Date**:
4 December 2009 (04.12.2009)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
61/121,418 10 December 2008 (10.12.2008) US
12/424,115 15 April 2009 (15.04.2009) US

(71) **Applicant** *(for all designated States except US)*: **COMMVAULT SYSTEMS, INC.** [US/US]; 2 Crescent Place, P.O. Box 900, Oceanport, NJ 07757 (US).

(72) **Inventor; and**
(75) **Inventor/Applicant** *(for US only)*: **NGO, David** [US/US]; 118 Borden Street, Shrewsbury, NJ 07702 (US).

(74) **Agent**: **ALTMAN, Daniel, E.**; Knobbe, Martens, Olson & Bear, LLP, 2040 Main Street, 14th Floor, Irvine, CA 92614 (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, 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, IS, JP, KE, KG, KM, 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, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (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, SE, SI, SK, SM,

[Continued on next page]

(54) **Title**: METHOD AND SYSTEM FOR MANAGING REPLICATED DATABASE DATA

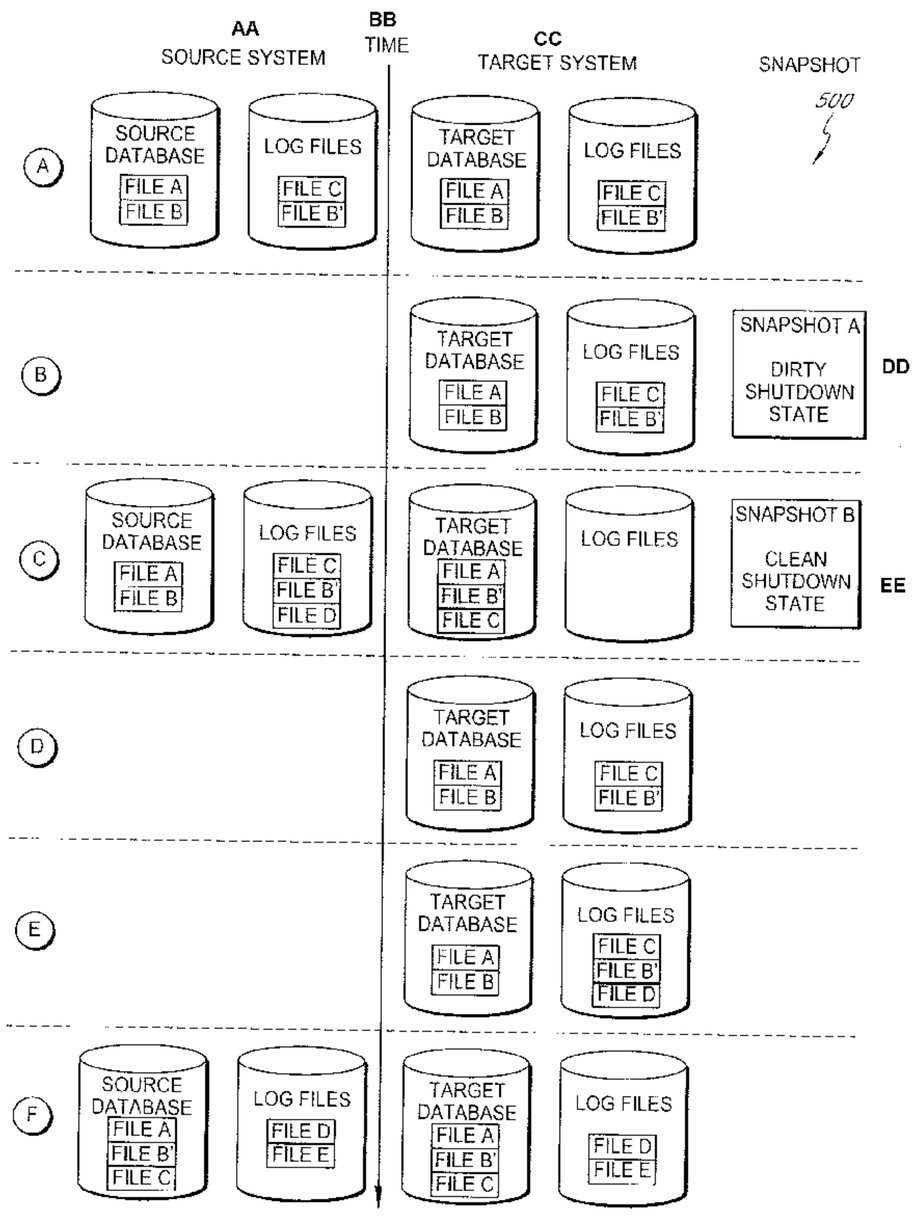


FIG. 5

(57) **Abstract**: Systems and methods for replicating database data and generating read-only copies of the replicated data in a clean shutdown state. For example, systems can include a tracking module (e.g., a filter driver) that monitors transactions from a database application to a source storage device to generate log entries having at least one marker indicating a known good state of the application. The systems further include a computer coupled to a target storage device comprising a database and log files. The computer processes the transactions, based on the log entries, to replicate data to the target storage device; performs a first snapshot on data stored in the database and log files; replays into the database data stored in the log files; performs another snapshot on the database; and reverts the database back to a state in which the database existed at the time of the first snapshot.

WO 2010/068570 A1

WO 2010/068570 A1

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

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

METHOD AND SYSTEM FOR MANAGING REPLICATED DATABASE DATA

BACKGROUND

Field of the Invention

[0001] The present disclosure relates to systems and methods for replicating data of one or more database applications.

Description of the Related Art

[0002] Electronic information has become an integral part of business operations such that many banks, insurance companies, brokerage firms, financial service providers, and a variety of other businesses rely on computer networks to store, manipulate, and display information that is constantly subject to change. Oftentimes, the success or failure of an important transaction may turn on the availability of electronic information that is both accurate and current. Accordingly, businesses seek reliable, cost-effective ways to protect and later access the information stored on their computer networks.

[0003] Many approaches to protecting data involve creating a copy of the data, such as backing up and/or replicating a database to one or more storage devices. Data shadowing and mirroring, or duplexing, provide for copying but can require lengthy amounts of time, consume valuable processing power and/or occupy large amounts of storage space for large databases. Moreover, such storage management systems can have a significant impact on the source or primary system.

[0004] To address these drawbacks, certain storage systems utilize snapshot techniques to preserve a read-only copy of database data. In general, a snapshot records the state of a storage device, database file system, or volume at a certain point in time. That is, the snapshot may be used to provide a point-in-time image of a live storage volume. Additional operations can then be performed using the snapshot copy without affecting the performance of the live volume.

[0005] In certain circumstances, however, a snapshot of a database does not necessarily provide easily accessible data. For instance, snapshots of data from

a MICROSOFT EXCHANGE database generated while EXCHANGE is online results in a copy of the data that is in a "dirty shutdown" state, which prevents the data from being read or otherwise accessed by standard application programming interfaces (APIs). Although, one option is to shut down EXCHANGE each time a snapshot is to be performed on the database, such an option is not practical as the shutdowns can be time-consuming and costly.

SUMMARY

[0006] In view of the foregoing, a need exists for improved systems and methods for database replication. For instance, there is a need for systems and methods for generating copies of database data in a clean shutdown state without taking the native database application offline. Moreover, a need exists for systems and methods that provide snapshots of database data in a useful condition without requiring the native application, or its accompanying APIs, for access to the database data.

[0007] In certain embodiments of the invention, systems and methods are disclosed for performing substantially continuous replication of a database, such as a MICROSOFT EXCHANGE database, and for providing usable read-only copies of the replicated database. For instance, snapshot systems and methods are disclosed that provide a snapshot of a replicated MICROSOFT EXCHANGE database that reflects data in a recoverable and clean shutdown state and that can be accessed or otherwise read using standard APIs without the need for the MICROSOFT EXCHANGE application program.

[0008] Such systems and methods allow known good replication copies to be viewed as copies of production volume data. For example, this technology, in certain embodiments, further allows a management component in the computing system to directly access, copy, restore, backup or otherwise manipulate the replication copies of production data as if the data was the production data of the source system, thereby improving various system performance characteristics such as access time, reducing memory requirements and reducing impact on source, or client, applications.

[0009] In certain embodiments, a method is disclosed for managing replicated data in a database system. The method comprises monitoring data transactions associated with a database application, the data transactions operative to write data to at least one source storage device. The method further includes copying the data to a target storage device based at least in part on the data transactions, wherein the target storage device comprises a target database and target transaction log files, and wherein said monitoring and copying is performed without shutting down the database application. The method also includes: (i) generating a first snapshot of at least a portion of the data stored in the target database and transaction log files; (ii) replaying, into the target database, data stored in the target transaction log files as one or more transaction logs; (iii) generating a second snapshot of at least a portion of the target database that is indicative of stored data from the database application in a recoverable state; and then (iv) reverting the target database back to a state in which the target database existed at the time of the first snapshot.

[0010] In certain embodiments, a system is disclosed for performing data management operations in a computer network environment. The system includes a database application configured to execute on a source computer and a first storage device coupled to the source computer to receive data transactions from the database application. The system also includes a module, such as a filter driver, configured to monitor the data transactions and to generate log entries based on the data transactions, at least one of the log entries having a marker indicative of a time of a known good state of the database application. The system further includes a second storage device comprising a target database and target transaction log files and a target computer coupled to the second storage device. The target computer is configured to: (i) process, based on the log entries, the data transactions to replicate data to the second storage device; (ii) perform a first snapshot operation on data stored in both the target database and the target transaction log files; (iii) replay into the target database data stored in the target transaction log files; (iv) perform a second snapshot operation on at least a portion of the target database;

and (v) revert the target database back to a state in which the target database existed at the time of the first snapshot.

[0011] In certain embodiments, a method is disclosed for copying data generated on a source database system in a computer network. The method comprises processing replication data indicative of data transactions generated by a database application executing on a source system, the database transactions being directed to operations on a source database of a source storage device; replaying the replication data on a target storage device to copy the data transactions to the target storage device, the target storage device comprising a target database and target transaction log files; creating a first read-only copy of at least a portion of data stored in the target database and the target transaction log files; committing to the target database transaction logs stored in the target transaction log files, said committing following said creating the first read-only copy; creating, following said committing, a second read-only copy of at least a portion of the target database, the second read-only copy being indicative of database data from the database application in a recoverable state; and reverting the target database to a state in which the target database existed at the time of said creating the first read-only copy.

[0012] In certain embodiments, a system is disclosed for performing data management operations in a computer network environment. The system includes means for processing replication data indicative of database operations generated by a database application executing on a source computer, the database operations being directed to data on a source storage device; means for replaying the replication data on a target storage device to copy data to the target storage device, the target storage device comprising a target database and transaction log files; means for performing a first snapshot operation on data stored in both the target database and the target transaction log files; means for replaying into the target database data stored in the transaction log files; said means for performing being further configured to perform a second snapshot operation on at least a portion of the target database following replaying of the data stored in the transaction log files, the second snapshot being indicative of database data from the database

application in a clean shutdown state; and means for reverting the target database back to a state in which the target database existed at the time of performing the first snapshot, wherein at least one of said means is executed on one or more computing devices.

[0013] For purposes of summarizing the disclosure, certain aspects, advantages and novel features of the inventions have been described herein. It is to be understood that not necessarily all such advantages may be achieved in accordance with any particular embodiment of the invention. Thus, the invention may be embodied or carried out in a manner that achieves or optimizes one advantage or group of advantages as taught herein without necessarily achieving other advantages as may be taught or suggested herein.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] FIGURE 1 illustrates a block diagram of a database replication system according to certain embodiments of the invention.

[0015] FIGURE 2 illustrates a block diagram of an exemplary embodiment of a database replication system configured to provide a snapshot of data in a clean shutdown state.

[0016] FIGURE 3 illustrates a flowchart of an exemplary embodiment of a database replication process usable by the database replication systems of FIGURES 1 and 2.

[0017] FIGURE 4 illustrates a flowchart of an exemplary embodiment of a double snapshot process of the data replication process of FIGURE 3.

[0018] FIGURE 5 illustrates a timeline of states of data during the database replication process of FIGURE 3, according to certain embodiments of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0019] Systems and methods disclosed herein provide for the offline access of a replicated database without needing to run the native database application. Moreover, disclosed systems and methods provide for the replication of

database data without shutting down the native database application to put the replicated data in a "clean shutdown" state.

[0020] In certain embodiments of the invention, systems and methods are disclosed for performing substantially continuous replication of a database, such as a MICROSOFT EXCHANGE database. For instance, certain embodiments provide for the generation of one or more snapshots of a replicated MICROSOFT EXCHANGE database, wherein the snapshot data allows for offline access to the replicated database. For instance, in certain embodiments, the replicated database can be read using JET or like APIs in place of slower APIs (e.g., MAPI) associated with MICROSOFT EXCHANGE. As a result, the direct reading of the database allows for much faster access to the replicated data.

[0021] In general, replication of database data can pose particular drawbacks. For example, standard replication of data from certain database applications can result in data that is in a recoverable state, wherein committed transactions in the database can be recovered in case of a crash. In such a state, particular computing operations of the application are complete to a point such that further operation, recovery and/or rolling back of the application can occur, based on the committed transaction data This point of referential integrity is generally referred to herein as a known good state of the application data.

[0022] However, without shutting down the database application, the data resulting therefrom is not in a clean shutdown state (i.e., transaction logs exist that are not committed to the database). Moreover, with MICROSOFT EXCHANGE databases, for instance, data in a dirty shutdown state is evidenced by a particular bit value and prevents standard APIs from accessing the data.

[0023] The features of the systems and methods will now be described with reference to the drawings summarized above. Throughout the drawings, reference numbers are re-used to indicate correspondence between referenced elements. The drawings, associated descriptions, and specific implementation are provided to illustrate embodiments of the invention and not to limit the scope of the disclosure.

[0024] In addition, methods and processes described herein are not limited to any particular sequence, and the blocks or states relating thereto can be performed in other sequences that are appropriate. For example, described blocks or states may be performed in an order other than that specifically disclosed, or multiple blocks or states may be combined in a single block or state.

[0025] Moreover, certain embodiments of the invention described herein can utilize one or more intelligent data replication processes and/or components that replicate application-specific data from a source system to a destination system. Certain examples of such processes and components are described in U.S. Patent Application No. 11/640,826, filed December 18, 2006, published as U.S. Patent Application Publication No. 2007-0185938 A1.

[0026] FIGURE 1 illustrates a block diagram of a database replication system 100 according to certain embodiments of the invention. As shown, the replication system 100 comprises a source system 102 capable of communicating with a target system 104 by sending and/or receiving data over a network 106. For instance, in certain embodiments, the target system 104 receives and/or stores a replicated copy of at least a portion of data, such as application-specific data, associated with the source system 102.

[0027] In certain embodiments, the source system 102 comprises one or more computing devices capable of processing data and can include, for example, a server computer, a workstation, a personal computer, a cell phone, a portable computing device, a handheld computing device, a personal digital assistant (PDA) or the like.

[0028] The illustrated network 106 advantageously comprises any means for communicating data between two or more systems or components. It certain embodiments, the network 106 comprises a computer network. For example, the network 106 may comprise a public network such as the Internet, virtual private network (VPN), token ring or TCP/IP based network, wide area network (WAN), local area network (LAN), an intranet network, point-to-point link, a wireless network, cellular network, wireless data transmission system, two-way cable system,

interactive kiosk network, satellite network, broadband network, baseband network, combinations of the same or the like. In embodiments wherein the source system 102 and target system 104 are part of the same computing device, the network 106 may represent a communications socket or other suitable internal data transfer path or mechanism.

[0029] As shown, the source system 102 comprises one or more applications 108 residing on and/or being executed by a computing device. For instance, the applications 108 may comprise software applications that interact with a user to process data and may include, for example, database applications (e.g., SQL applications), word processors, spreadsheets, financial applications, management applications, e-commerce applications, browsers, combinations of the same or the like.

[0030] The source system 102 further comprises one or more processes, such as filter drivers 110, that interact with data (e.g., production data) associated with the applications 108. For instance, the filter driver 110 can comprise a file system filter driver, an operating system driver, a filtering program, a data trapping program, an application, a module of the application(s) 108, an API, or other like software module or process that, among other things, monitors and/or intercepts particular application requests targeted at a database, a file system, another file system filter driver, a network attached storage ("NAS"), a storage area network ("SAN"), mass storage and/or other memory or raw data. In some embodiments, the filter driver 110 may reside in the I/O stack of the application 108 and may intercept, analyze and/or copy certain data traveling from the application 108 to storage.

[0031] In certain embodiments, the filter driver 110 can intercept data modification operations that include changes, updates and new information (e.g., data writes) with respect to the application(s) 108 of interest. For example, the filter driver 110 may locate, monitor and/or process one or more of the following with respect to a particular application 108, application type or group of applications: data management operations (e.g., data write operations, file attribute modifications), logs or journals (e.g., NTFS change journal), configuration files, file settings, control

files, other files used by the application 108, combinations of the same or the like. In certain embodiments, such data may also be gathered from files across multiple storage systems within the source system 102. Furthermore, the filter driver 110 may be configured to monitor changes to particular files, such as files identified as being associated with data of the applications 108.

[0032] In certain embodiments, multiple filter drivers 110 may be deployed on the source system 102, each filter driver 110 being dedicated to data of a particular application 108. In such embodiments, not all information associated with the source system 102 may be captured by the filter drivers 110 and, thus, the impact on system performance may be reduced. In other embodiments, the filter driver 110 may be suitable for use with multiple application types and/or may be adaptable or configurable for use with multiple applications 108.

[0033] The illustrated source system 102 further comprises one or more source storage devices 120. In certain embodiments, the source storage device(s) 120 are configured to store production data associated with one or more of the applications 108. The source storage device 120 may include any type of media capable of storing data. For example, the source storage device 120 may comprise magnetic storage (such as a disk or a tape drive) or other type of mass storage. In certain embodiments, the source storage device 120 may be internal and/or external to (e.g., remote to) the computing device(s) having the applications 108 and the filter drivers 110.

[0034] As illustrated in FIGURE 1, the source storage devices 120 further comprise at least one database 122 and transaction log files 124. In particular, the transaction log files 124 can comprise a set of changes, such as, for example, insertions, deletions and updates received from the application(s) 108 that are to be applied to the data in the database 122.

[0035] In certain embodiments, the transaction log files 124 are stored on at least one dedicated disk so that the logs are not affected by any disk failures that can potentially corrupt the database 122. For instance, the transaction log files 124 can be stored on a high-performance disk while the database 122 is stored on one

or more slower disks. In other embodiments, the transaction log files 124 and the database 122 can be maintained on the same storage medium.

[0036] As further illustrated in FIGURE 1, the target system 104 comprises a replication module 128 that communicates with one or more target storage devices 130. In certain embodiments, the target system 104 comprises any computing device capable of processing data and includes, for example, a server computer, a workstation, a personal computer or the like.

[0037] In certain embodiments, the replication module 128 is configured to monitor and/or manage the copying of data from the source system 102 to the target system 104, such as data obtained by the filter drivers 110. For example, the replication module can receive one or more log files or entries, data transactions, or other like replication data indicative of the data transactions or operations being generated by the application 108 to modify data stored on the source storage device(s) 120. In yet other embodiments, the replication module 128 is a "dumb" server or terminal that receives and executes instructions from the source system 102.

[0038] The target storage device(s) 130 may include any type of media capable of storing data, such as replication data sent from the source system 102. For example, the target storage device 130 may comprise magnetic storage (such as a disk or a tape drive) or other type of mass storage. In certain embodiments, the target storage device 130 may be internal and/or external to the computing device(s) having the replication module 128.

[0039] In certain embodiments, the source storage device 120 and/or the target storage device 130 may be implemented as one or more storage "volumes" that include physical storage disks defining an overall logical arrangement of storage space. For instance, disks within a particular volume may be organized as one or more groups of redundant array of independent (or inexpensive) disks (RAID). In certain embodiments, either or both of the storage devices 120, 130 can include multiple storage devices of the same or different media.

[0040] As shown in FIGURE 1, the target storage device(s) 130 at least comprise a database 132 and transaction log files 134. In particular, the target

database 132 is generally synchronized with the source database 122. Likewise, the target transaction log files 134 can comprise a replication of those log files present in the transaction log files 124 of the source storage device 120. As discussed above, in certain embodiments, the database 132 and the transaction log files 134 are advantageously maintained on separate media such that the transaction log files 134 can be stored on a relatively high-performance storage medium. In yet other embodiments, the database 132 and transaction log files 134 are stored in different volumes of the same disk.

[0041] As further shown, the replication system 100 is structured to generate read-only copies 136 of the replicated data stored in the target storage device(s) 130. For instance, such copies 136 can be generated according to one or more schedules, storage policies (e.g., user- or system-defined), or the like. In certain embodiments, the replication module 128 coordinates the generation of the copies 136. For example, the replication module 128 can comprise a data agent module executing on the replication system 100 that manages read-only copies (e.g., snapshots) of the replicated data. In yet other embodiments, a storage manager, a stand-alone application, or the like, manages the generation of the read-only copies 136.

[0042] In certain embodiments, the read-only copies 136 advantageously comprise a plurality of snapshots. For instance, the snapshots can reflect point-in-time copies of the database data. In certain embodiments, the snapshots allow for access to and/or manipulation of database data without affecting the production data on the source storage device(s) 120. Moreover, in certain embodiments, the snapshots record database data in a clean shutdown mode and allow for offline access of the data without the need for the native database application 108.

[0043] In certain embodiments, the source system 102 communicates with the associated target system 104 to verify that the two systems are synchronized. For instance, the source system 102 may receive from the target system 104 an identification (e.g., unique serial number) of a transaction log last committed by the target system 104. The source system 102 may then compare the received identification with state of the data in the source storage device(s) 120.

[0044] FIGURE 2 illustrates further details of a replication system 200 in accordance with certain embodiments of the invention. In particular, FIGURE 2 illustrates additional details of an embodiment of the replication system 100 of FIGURE 1. Thus, for ease of reference and description, elements of the replication system 200 of FIGURE 2 will not be redescribed in detail if they were described above. Rather, the elements in the embodiment of FIGURE 2 will be given a reference numeral that retains the same last two digits as the reference numeral used in the embodiment of FIGURE 1, and the last two digits will be preceded with a numeral "2." Thus, the replication system 200 generally corresponds to the replication system 100 with certain differences that will be illuminated in the following discussion.

[0045] As shown, the replication system 200 comprises a source system 202 that communicates with a target system 204 via a network 206. In particular, the target system 204 is structured to replicate database data originating from and/or stored by the source system 202.

[0046] The illustrated source system 202 comprises a database application 208 that interacts with one or more users or programs to generate production data. For instance, the database application 208 can comprise one or more applications that require transactions to be played or committed in order for the data of the application 208 to be at consistent point or state.

[0047] For exemplary purposes, the database application 208 will often be described hereinafter with reference to a MICROSOFT EXCHANGE environment. In yet other embodiments, other types of database applications and/or collaboration applications can be used, such as, for example, ACTIVE DIRECTORY, ORACLE, other JET-based applications or the like.

[0048] With continued reference to the source system 202, a filter driver 210 tracks file changes being generated by the application 208. For example, the filter driver 210 can be configured to populate data relating to transactions of the database application 208 in source logs to be replicated to the target system 204.

[0049] In certain embodiments, the filter driver 210 may be deployed in the stack as an I/O buffer and/or process in the data path between the database

application 208 and a memory 212. In such embodiments, the filter driver 210 may intercept, snoop, supervise, trap, process or otherwise be cognizant of some or all operations (e.g., data modification operations, file modification operations, read operations and the like) from the database application 208 to its associated location(s) on the source storage device(s) 220.

[0050] In embodiments of the invention having multiple database applications 208, multiple filter drivers 210 can be used such that each filter driver 210 corresponds to a single database application 208. In such embodiments, data relating to each database application 208 of interest can be written to a particular log file established for that application. In yet other embodiments, a single filter driver 210 can communicate with multiple database applications 208.

[0051] In certain embodiments, the transactions from the database application 208 are preferably stored in the memory 212 after or while being logged by the filter driver 210. A database engine 214 then commits the transactions stored in the memory 210 to disk (source storage device(s) 220).

[0052] The source storage device(s) 220 comprise one or more media for storing production data relating to the database application 208. As shown, the source storage device 220 includes a database or information store 222 for storing the committed transaction data from the database application 208. The source storage device(s) 220 further include transaction log files 224 that represent a sequence of files to keep a secure copy on disk of the data (e.g., uncommitted transaction logs) in the memory 212.

[0053] As described above with reference to FIGURE 1, the replication system 200 is configured to replicate data, such as through a continuous data replication process, to target storage device(s) 230 of the target system 204. To facilitate and/or coordinate such replication, the target system 204 includes a replication module 228.

[0054] In certain embodiments, the replication module 228 comprises and/or communicates with one or more replay threads, processes or routines that populate data to the target storage device(s) 230 during the replication process. For instance, the replication module 228 can replay data from log files that are received

from the source system 202. In certain embodiments, the target storage device(s) 230 comprise one or more databases 232 and transaction log files 234 that store a copy of the data residing in, respectively, the source database(s) 222 and the transaction log files 224.

[0055] The target system 204 is further configured to generate one or more snapshots 236 of data stored in the target storage device(s) 230. In certain embodiments, the snapshot 236 comprises a read-only copy of the data stored in the database 232 and the transaction log files 234, as discussed in more detail above.

[0056] In certain embodiments, the replication system 200 can optionally include one or more additional components for further coordinating and/or facilitating replication between the source system 202 and the target system 204. For instance, FIGURE 2 further illustrates the source system 202 comprising an optional data agent 216.

[0057] In certain embodiments, the data agent 216 comprises a module responsible for performing data and/or storage tasks related to the source system 202. For example, the data agent 216 may manage and/or coordinate the compilation of and/or transferring of replication data from the source system 202. In other embodiments, the data agent 216 may provide archiving, migrating, and/or recovery of system data.

[0058] In certain embodiments, the source system 202 comprises a plurality of data agents 216, each of which performs data management operations related to data associated with each database application 208. In such embodiments, the data agent 216 may be aware of the various logs, files, folders, registry files and/or system resources that are impacted by a particular database application 208.

[0059] In certain embodiments, the data agent 216 is configured to perform data management operations in accordance with one or more "storage policies" or other preferences. In certain embodiments, a storage policy includes a data structure or other information having a set of preferences and/or other storage criteria for performing a storage operation. The preferences and storage criteria

may include, but are not limited to, information regarding storage locations and/or timing, relationships between system components, network pathways, retention policies, data characteristics, compression or encryption requirements, preferred system components, combinations of the same or the like.

[0060] In certain further embodiments, the replication system 200 can optionally include a storage manager 238 that communicates with the source system 202 and the target system 204. In certain embodiments, the storage manager 238 directs the performance of one or more storage operations and, in particular, the replication of data from the source system 202 to the target system 204. In further embodiments, the storage manager 238 may perform one or more of the operations or functions described above with respect to the data agent 216 and/or the replication module 228. For instance, the storage manager 238 may direct and/or coordinate the performance of one or more storage operations on the replicated data (e.g., snapshots of the replicated data) of the target storage device(s) 230.

[0061] In certain embodiments, the storage manager 238 maintains an index (not shown), such as a cache, for storing information relating to: logical relationships and associations between components of the replication system 200, user preferences, management tasks, and/or other useful data. For example, the storage manager 238 may use its index to track the location and timestamps of one or more snapshots of the replicated data.

[0062] The storage manager 238 may also use its index to track the status of data management operations to be performed, storage patterns associated with the system components such as media use, storage growth, network bandwidth, Service Level Agreement ("SLA") compliance levels, data protection levels, storage policy information, storage criteria associated with user preferences, retention criteria, storage operation preferences, and/or other storage-related information. The index may typically reside on the storage manager's hard disk and/or other database.

[0063] In certain embodiments, the storage manager 238 further communicates with a database (not shown) for storing system management

information relating to the replication of data. For instance, the storage manager database may be configured to store storage and/or restore policies, user preferences, the status or location of system components or data, combinations of the same and the like. In yet other embodiments, the storage manager database may be configured to store information described above with respect to the storage manager index. In yet other embodiments, at least a portion of the storage manager index may be stored on the storage manager database.

[0064] In other embodiments, the storage manager 238 may alert the user or system when a particular resource of the replication system 200 is unavailable or congested or when components are unavailable due to hardware failure, software problems, or other reasons. In certain embodiments, the storage manager 238 may utilize replication system data to suggest solutions to such problems when they occur or even before they occur. For example, the storage manager 238 might alert the user that a storage device in the target system 204 is full or otherwise congested, and then suggest, based on job and data storage information contained in its index cache, an alternate storage device. In yet further embodiments, the storage manager 238 or other system component may take action to remedy the problem at issue. For example, the storage manager 238 may perform load balancing, error correction, or the like, based on information received regarding the target system 204.

[0065] In certain embodiments, the storage manager 238 may include other components and/or modules. For example, the storage manager 238 may include a jobs agent module (not shown) that monitors the status of storage operations that have been performed, that are being performed, or that are scheduled to be performed in the replication system 200.

[0066] Moreover, the storage manager 238 may include an interface agent module (not shown). In certain embodiments, the interface agent module may provide presentation logic, such as a graphical user interface ("GUI"), an API, or other interface by which users and system processes may be able to retrieve information about the status of storage operations and issue instructions to the replication system 200 regarding the performance of storage operations. For

example, a user may modify the schedule of a number of pending snapshot copies or other types of copies. As another example, a user may use the GUI to view the status of all storage operations currently pending in the replication system 200 or the status of particular components in the replication system 200.

[0067] Additional details of storage manager modules useful with embodiments of the replication systems 100, 200 described herein are disclosed in U.S. Patent Application No. 09/354,063, filed July 15, 1999, now U.S. Patent No. 7,389,311.

[0068] FIGURE 3 illustrates a flowchart of an exemplary embodiment of a database replication process 300. In certain embodiments, the process 300 creates a crash consistent copy of database data in a clean shutdown state such that the data can be read or otherwise utilized without needing the native database application or its accompanying APIs. In certain embodiments, the replication process 300 can be performed by the database replication systems 100, 200 of FIGURES 1 and 2. For exemplary purposes, the replication process 300 will be described hereinafter with respect to the components of the replication system 200 of FIGURE 2.

[0069] The replication process 300 begins at Block 305, wherein the database application 208 is quiesced. For example, in certain embodiments, the Volume Shadow Copy Service (VSS) offered by MICROSOFT can be used to temporarily quiesce write requests from the database application 208, such as MICROSOFT EXCHANGE, to the source storage device 220. For instance, in such embodiments, the data agent 216 can insert a marker in the application data (e.g., transaction logs) that indicates the data is in a recoverable (e.g., crash consistent, stable) state.

[0070] At Block 310, the replication module 228 suspends the replay threads on the target system 204 that are replicating the data to the target storage device 230. In certain embodiments, such suspension occurs based on the threads' detection of the markers inserted by the VSS service. At this point, the target storage device(s) 230 and the source storage device(s) 220 are preferably

synchronized and/or include data that is in a crash recoverable state, although the data is not in a "clean shutdown" state.

[0071] With the replay threads suspended, the replication module 228 initiates a first snapshot of the target volume, including the database 232 and the transaction log files 234 (Block 315). In certain embodiments, the first snapshot includes a read-only copy of the database application data in a recoverable state. However, as discussed above, this data is generally not in a clean shutdown state and does not allow for access of the data without the use of the native application (e.g., MICROSOFT EXCHANGE) to interpret the data.

[0072] At Block 320, the copy of the uncommitted transaction logs stored in the target transaction log files 234 is replayed into the target database 232. In certain embodiments, such replaying is performed through the execution of one or more JET APIs. After this point, the copy of the database application 208 data in the target database 232 is advantageously in a clean shutdown state since the pending transaction logs have been committed. That is, the copy of the data in the database 232 is in a state that can be read offline without the use of the native database application 208.

[0073] At Block 325, the replication module 228 initiates a second snapshot of the target volume. Because the target transaction logs have been replayed into the target database 232, the second snapshot advantageously comprises a read-only copy of application data in both a recoverable and clean shutdown state.

[0074] However, with the replaying of the copy of the transaction logs at Block 320, the data on the target storage device(s) 230 becomes out of sync with the production data stored on the source storage device(s) 220. Thus, prior to resuming replication between the source storage device(s) 220 and the target storage device(s) 230, the replication process 300 reverts the target volume back to the state of the first snapshot (Block 330). In certain embodiments, this revert process is performed by taking a difference (e.g., file changes that have been cached) of the first and second snapshots, as is discussed in more detail with respect to FIGURE 4.

[0075] Following Block 330, the replication module 228 thaws the replay threads and allows the replication between the source and target storage devices 220, 230 to resume (Block 335).

[0076] Although the database replication process 300 has been described with reference to a particular arrangement, other embodiments of the invention can include more, fewer or different blocks or states. For instance, in other embodiments, instead of snapshots, other types of read-only copies can be performed on the data in the target storage device(s) 230.

[0077] In yet other embodiments, at Block 305, the data agent 216 and/or the filter driver 210 can be advantageously configured to pause, or quiesce, the database application 208 during data replication. For instance, the data agent 216 may cause the application 208 to temporarily suspend data management operations to the source storage device 220 once the application 208 reaches a known "good," "stable" or "recoverable" state.

[0078] In certain embodiments, the data agent 216 instructs the quiescing of the application 208 through an application programming interface (API). When the application 208 has placed itself in a known good state, the application 208 may send an acknowledgment to the data agent 216.

[0079] In certain embodiments, once the data management operations are suspended, the filter driver 210 and/or data agent 216 then inserts a logical marker or tag in the source log file denoting that a "consistency point" or "consistency recovery point" has been reached. In some embodiments, the consistency point indicates the time at which the application 208 is at a known good state.

[0080] Moreover, in certain embodiments, the target system 204 is further capable of performing one or more data management operations, such as, for example, storage operations (e.g., backup), search operations, data classification, combinations of the same or the like on the replicated data at certain consistency points. Performing data management operations on the replicated data allows for the processing of copies of application data without significantly impacting the resources of the source system. Furthermore, when copying the replicated data at

consistency points, the copied data presumably represents a known good state of the application.

[0081] In certain embodiments of the invention, at Block 305 the application 208 is periodically quiesced based on particular criteria. For instance, the quiescing of the application 208 may be based on one or more system- or user-defined preferences (e.g., every five minutes). The periodic quiescing of the application 208 may be based on the desired frequency of performing replication, backup or other data modification operations on the subject data. For instance, applications 208 dealing with data-sensitive information may necessitate more frequent quiescing (and creation of consistency points) than other types of applications.

[0082] In yet other embodiments, quiescing of the application 108 may be performed based on an automatic reporting procedure. For instance, a module of the replication system 200 may be configured to gather, receive and/or analyze information associated with a failure rate and/or health of applicable servers. Additional details of such status monitoring are provided in U.S. Patent Application No. 11/130,619, filed May 2, 2005, now published as U.S. Patent Application Publication No. 2006-0053261 A1.

[0083] FIGURE 4 illustrates a flowchart of an exemplary embodiment of a double snapshot process 400 that may be used when performing certain blocks (e.g., Blocks 315 to 330) of the data replication process 300 of FIGURE 3. For exemplary purposes, the double snapshot process 400 will be described hereinafter with respect to the components of the replication system 200 of FIGURE 2.

[0084] At Block 405, the process 400 performs a first snapshot of the target volume, including the target database 232 and the target transaction log files 234. Following the first snapshot, the transaction logs are replayed into the target database 232 to put the copy of the data in a clean shutdown state. During the replay of the transaction logs (e.g., by a JET API), the overwritten bits in the target database 232 can be moved to a first cache location (Block 410).

[0085] The process 400 then performs a second snapshot of the target volume, wherein the target volume represents the application data in a clean shutdown state (Block 415). Following the second snapshot, the process 400 reverts the target volume back to the state in which it existed at the time of the first snapshot. To do so, at Block 420, the new bits added between the first and second snapshots (i.e., during the replaying of the transaction logs) are moved to a second cache location (Block 425). The overwritten bits stored in the first cache location are then moved back to the target volume (Block 425).

[0086] At Block 430, the second snapshot is preserved as a consistent recovery point in a clean shutdown state. In certain embodiments, a customized driver is used to preserve the snapshot (e.g., the snapshot is not deleted).

[0087] In certain embodiments, the first and second cache locations can be on different storage devices or volumes. In yet other embodiments, the first and second cache locations can be located on the same storage device.

[0088] FIGURE 5 illustrates an exemplary timeline 500 of states of data during the database replication process 300 of FIGURE 3, according to certain embodiments of the invention. In particular, the timeline 500 includes a simplified representation of the states of files on a source system (e.g., source storage device 220) and a target system (e.g., target storage device 230) during replication.

[0089] As shown, at State A, the source database includes File A and File B and the source log files include a transaction relating to the creation of File C and the modification of File B (i.e., File B'). With reference to the target system, the target database and the target log files are synchronized with the source system.

[0090] At State B, a first snapshot (i.e., Snapshot A) is taken of the target system. As discussed above with reference to Block 315 of the replication process 300, the application data from the target database is in a dirty shutdown state.

[0091] At State C, an additional transaction log for the creation of a File D has been added to the source log files. Moreover, the target transaction logs (Files B' and C) have been replayed from the target log files to the target database (see, e.g., Block 320). Thus, the target database includes File A, File B' and File C. At this point, the data in the target database is in a clean shutdown state, and no

outstanding transaction logs are left in the target log files. As further illustrated, the system then takes a second snapshot (Snapshot B) of at least the data in the target database to create a consistent, recoverable copy of the database application data that can be accessed and utilized without the native database application which created the data (see, e.g., Block 325).

[0092] At State D, the target database is reverted back to the state in which it existed at the time Snapshot A was taken (see, e.g., Block 330), and the target log files are repopulated with transactions relating to Files B' and C. Moreover, at State E, a transaction log relating to File D is replicated from the source system to the target system.

[0093] At State F, the transaction logs in both the source and target log files relating to File B' and File C are replayed, respectively, into the source and target databases. Moreover, an additional transaction log relating to File E is added to the source log files and replicated to the target system such that the source and target systems are re-synchronized.

[0094] In certain embodiments of the invention, data replication systems and methods disclosed herein may be used in a modular storage management system, embodiments of which are described in more detail in U.S. Patent No. 7,035,880, issued April 5, 2006. For example, the data replication system may be part of a storage operation cell that includes combinations of hardware and/or software components directed to performing storage operations on electronic data. Exemplary storage operation cells usable with embodiments of the invention include CommCells as embodied in the QNet storage management system and the QiNetix storage management system by CommVault Systems, Inc. (Oceanport, New Jersey), and as further described in U.S. Patent Application No. 10/877,831, filed June 25, 2004, now published as U.S. Patent Application Publication No. 2005-0033800 A1.

[0095] Systems and modules described herein may comprise software, firmware, hardware, or any combination(s) of software, firmware, or hardware suitable for the purposes described herein. Software and other modules may reside

on servers, workstations, personal computers, computerized tablets, PDAs, and other devices suitable for the purposes described herein. Software and other modules may be accessible via local memory, via a network, via a browser, or via other means suitable for the purposes described herein. Data structures described herein may comprise computer files, variables, programming arrays, programming structures, or any electronic information storage schemes or methods, or any combinations thereof, suitable for the purposes described herein. User interface elements described herein may comprise elements from graphical user interfaces, command line interfaces, and other interfaces suitable for the purposes described herein.

[0096] Embodiments of the invention are also described above with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, may be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the acts specified in the flowchart and/or block diagram block or blocks.

[0097] These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to operate in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the acts specified in the flowchart and/or block diagram block or blocks. The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operations to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus

provide steps for implementing the acts specified in the flowchart and/or block diagram block or blocks.

[0098] While certain embodiments of the inventions have been described, these embodiments have been presented by way of example only, and are not intended to limit the scope of the disclosure. Indeed, the novel methods and systems described herein may be embodied in a variety of other forms. The scope of the claims should not be limited by particular embodiments set forth herein, but should be construed in a manner consistent with the specification as a whole.

CLAIMS:

1. A method for managing replicated data in a database system, the method comprising:

monitoring a plurality of data transactions associated with a database application, the plurality of data transactions operative to write data to at least one source storage device;

copying the data to at least one target storage device based at least in part on the plurality of data transactions, wherein said at least one target storage device comprises a target database and target transaction log files, wherein said monitoring and copying is performed without shutting down the database application;

generating a first snapshot of at least a portion of the data stored in the target database and the target transaction log files, the first snapshot being indicative of stored data from the database application in a dirty shutdown state that prevents the first snapshot from being read or otherwise accessed by applications other than the database application, a stored indication of the dirty shutdown state being associated with the first snapshot;

replaying into the target database, following the first snapshot, data stored in the target transaction log files as one or more transaction logs such that the data stored in the target database following said replaying becomes out of sync with the data stored on the source storage device;

generating, following said replaying, a second snapshot of at least a portion of the target database, the second snapshot being indicative of stored data from the database application in a recoverable and clean shutdown state; and

reverting, following said generating the second snapshot, the target database back to a state in which the target database existed at the time of said generating the first snapshot, wherein the data stored in the target database following said reverting is re-synchronized with the data stored on the source storage device, wherein the second snapshot can be read or otherwise accessed without the use of the database application.

2. The method of claim 1, wherein said generating the first snapshot is performed based on a marker in the copied data indicating a known good state of the database application.

3. The method of claim 2, further comprising:

quiescing the database application at a time of the known good state of the database application; and

inserting the marker in the copied data.

4. The method of claim 1, wherein said copying the data further comprises transmitting between the at least one source storage device and the at least one target storage device a plurality of log entries indicative of the plurality of data transactions associated with the database application.

5. The method of claim 1, further comprising:

moving to a first memory location data overwritten in the target database during said replaying;

moving to a second memory location data added to the target database during said replaying; and

wherein said reverting comprises moving the overwritten data back to the target database following said generating the second snapshot.

6. The method of claim 1, further comprising suspending said copying prior to said generating the first snapshot.

7. The method of claim 6, wherein said suspending is performed at a time at which the data stored on the target database is in a known good state.

8. The method of claim 1, further comprising suspending said copying according to one or more storage policies defined by a user.

9. The method of claim 1, further comprising synchronizing the source database and the target database following said reverting.

10. The method of claim 1, further comprising reading the second snapshot with one or more application programming interfaces while the database application is offline with respect to the data in the second snapshot.

11. A system for performing data management operations in a computer network environment, the system comprising:

a database application configured to execute on a source computer;

a first storage device coupled to the source computer to receive data transactions from the database application;

at least one module configured to monitor the data transactions and to generate a plurality of log entries based on the data transactions, at least one of the plurality of log entries having a marker indicative of a time of a known good state of the database application;

a second storage device comprising a target database and target transaction log files; and

a target computer coupled to the second storage device and configured to:

process, based on the plurality of log entries, the data transactions to replicate data to the second storage device,

perform a first snapshot operation on data stored in both the target database and the target transaction log files to create a first snapshot, the first snapshot being indicative of stored data from the database application in a dirty shutdown state that prevents the first snapshot from being read or otherwise accessed by applications other than the database application, a

stored indication of the dirty shutdown state being associated with the first snapshot,

following the first snapshot operation, replay into the target database data stored in the target transaction log files such that the data stored in the target database following the replay becomes out of sync with the data stored on the first storage device,

following said replaying, perform a second snapshot operation on at least a portion of the target database to create a second snapshot, the second snapshot being indicative of stored data from the database application in a recoverable and clean shutdown state, and

revert, following the performance of the second snapshot, the target database back to a state in which the target database existed at the time of said performing the first snapshot wherein the data stored in the target database following said reverting is re-synchronized with the data stored on the first storage device, wherein the second snapshot can be read or otherwise accessed without the use of the database application.

12. The system of claim 11, wherein the target computer further comprises a replication module having a plurality of threads configured to replay the data transactions to replicate the data to the second storage device.

13. The system of claim 11, wherein the target computer comprises at least one application programming interface configured to commit to the target database data stored in the transaction log files.

14. The system of claim 11, wherein the at least one module comprises a filter driver.

15. The system of claim 11, wherein performing the second snapshot operation produces a read-only copy of the target database data, the read-only copy being

indicative of database data that can be read without the use of the database application.

16. The system of claim 11, wherein the target computer is further configured to:

store data overwritten in the target database during the replaying of the data stored in the transaction log files into the target database; and

store data added to the target database during the replaying of the data stored in the transaction log files into the target database.

17. A system for performing data management operations in a computer network environment, the system comprising:

a target computer comprising computer hardware, the target computer coupled to a source computer, wherein the source computer has a native database application executing thereon and is coupled to at least one first storage device storing a source database including data generated by the native database application;

at least one second storage device coupled to the target computer, wherein the target computer is configured to:

generate a target database on the second storage device, the target database including a replicated version of at least some of the data in the source database;

perform a first copy operation to generate a first copy of at least a portion of the target database, wherein performing the first copy operation puts the target database into an unclean shutdown state;

subsequent to the first copy operation, commit a plurality of data transactions into the target database, the plurality of data transactions generated by the native database application and represented by a plurality of corresponding log entries, wherein the committing of the plurality of data transactions into the target database occurs prior to the plurality of data transactions being committed into the source database, wherein the data in

the source database and the data in the target database are out of sync following the committing of the plurality of data transactions into the target database, and wherein the committing of the plurality of data transactions into the target database puts the target database into a clean shutdown state;

subsequent to the committing of the plurality of data transactions into the target database, perform a second copy operation to generate a second copy of at least a portion of the target database which includes data corresponding to the plurality of committed data transactions, wherein the second copy is generated in the clean shutdown state; and

revert the target database back to the unclean shutdown state in which the target database existed at the time the first copy operation was performed,

wherein the clean shutdown state allows for offline access of the data in the respective target database or second copy without the need for the native database application, and

wherein the unclean shutdown state requires the native database application for offline access of the data in the respective target database or second copy.

18. The system of claim 17, wherein at least one of the plurality of log entries comprises a marker indicative of a time of a known good state of the native database application.

19. The system of claim 17, wherein the first copy operation and the second copy operation comprise point-in-time snapshot operations.

20. The system of claim 17, wherein the target computer further comprises a replication module having a plurality of threads configured to commit the plurality of data transactions.

21. The system of claim 17, wherein the target computer comprises at least one application programming interface configured to commit the plurality of data transactions represented by the log entries.

22. The system of claim 17, wherein a filter driver is configured to monitor data transactions generated by the native database application and generate the plurality of log entries.

23. The system of claim 17, wherein the second copy is a read-only copy of the target database.

24. The system of claim 17, wherein the generation of the target database occurs without shutting down the native database application.

25. The system of claim 17, wherein said first and second copies comprise point-in-time images.

26. A method for managing data in a database system, the method comprising:

storing log entries in one or more target storage devices, wherein said log entries represent uncommitted data transactions generated by a native database application and associated with a source database stored on at least one source storage device;

prior to the data transactions being committed on either of the source data base and the target database, generating a first copy of at least a portion of a target database stored on the one or more target storage devices, wherein generating the first copy puts the target database into an unclean shutdown state;

committing into the target database, subsequent to said generating the first copy, the data transactions represented by the log entries, wherein said committing into the target database occurs prior to the data transactions being committed into the source database, and wherein the data in the source database and the data in the

target database are out of sync following said committing into the target database, and wherein said committing into the target database puts the target database into a clean shutdown state;

generating, subsequent to said committing into the target database, a second copy of at least a portion of the target database which includes data corresponding to the committed data transactions, wherein the second copy is generated in the clean shutdown state; and

reverting the target database back to the unclean shutdown state in which the target database existed at the time of said generating the first copy,

wherein the clean shutdown state allows for offline access of the data in the respective target database or second copy without the need for the native database application, and

wherein the unclean shutdown state requires the native database application for offline access of the data in the respective target database or second copy.

27. The method of claim 26, wherein said generating the first copy is performed based on a marker in one of the log entries indicating a known good state of the native database application.

28. The method of claim 27, further comprising:

quiescing the native database application at a time of the known good state of the native database application; and

inserting the marker in the one of the log entries.

29. The method of claim 26, wherein the log entries are transmitted between the at least one source storage device and the one or more target storage devices.

30. The method of claim 26, wherein the first copy and the second copy comprise point-in-time snapshot copies.

31. The method of claim 26, wherein said storing is performed without shutting down the native database application.

32. The method of claim 26, wherein said first and second copies comprise point-in-time images.

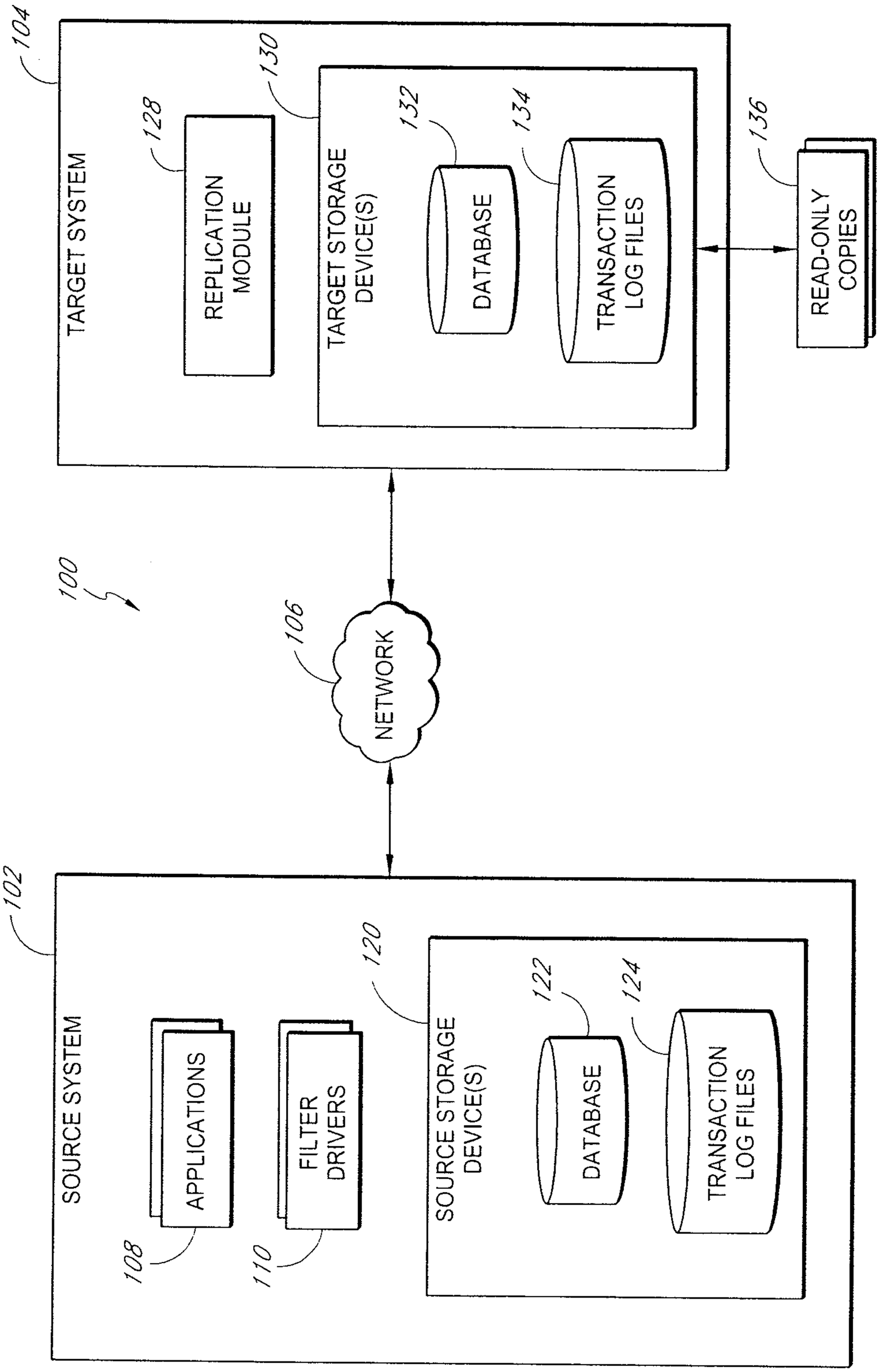

100
102
SOURCE SYSTEM
APPLICATIONS
108
FILTER DRIVERS
110
SOURCE STORAGE DEVICE(S)
120
DATABASE
122
TRANSACTION LOG FILES
124
NETWORK
106
104
TARGET SYSTEM
REPLICATION MODULE
128
TARGET STORAGE DEVICE(S)
130
DATABASE
132
TRANSACTION LOG FILES
134
READ-ONLY COPIES
136

FIG. 1

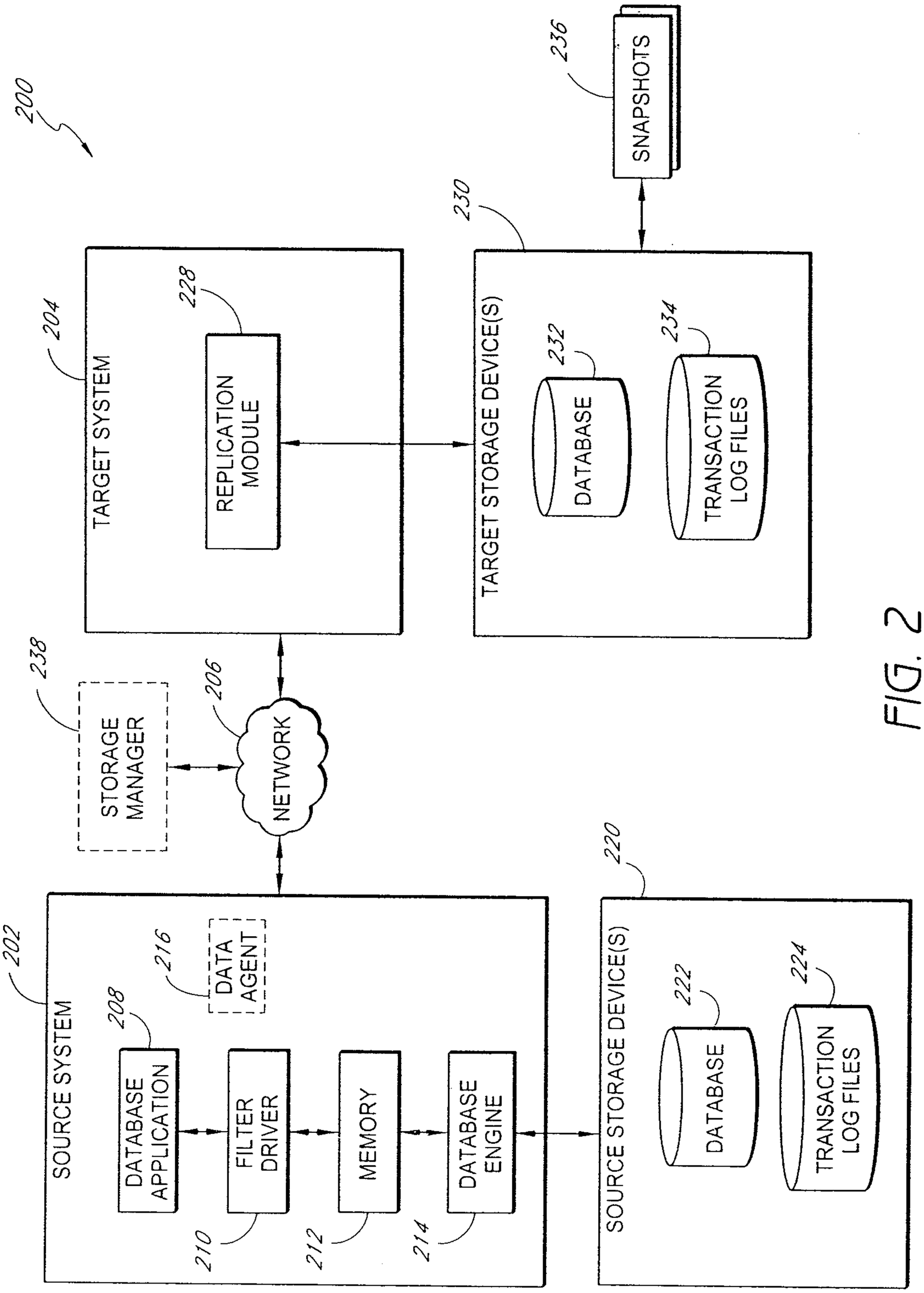

200
SNAPSHOTS
236
230
TARGET SYSTEM
204
228
REPLICATION MODULE
TARGET STORAGE DEVICE(S)
232
DATABASE
234
TRANSACTION LOG FILES
238
STORAGE MANAGER
206
NETWORK
SOURCE SYSTEM
202
208
DATABASE APPLICATION
216
DATA AGENT
FILTER DRIVER
MEMORY
DATABASE ENGINE
210
212
214
220
SOURCE STORAGE DEVICE(S)
222
DATABASE
224
TRANSACTION LOG FILES

FIG. 2

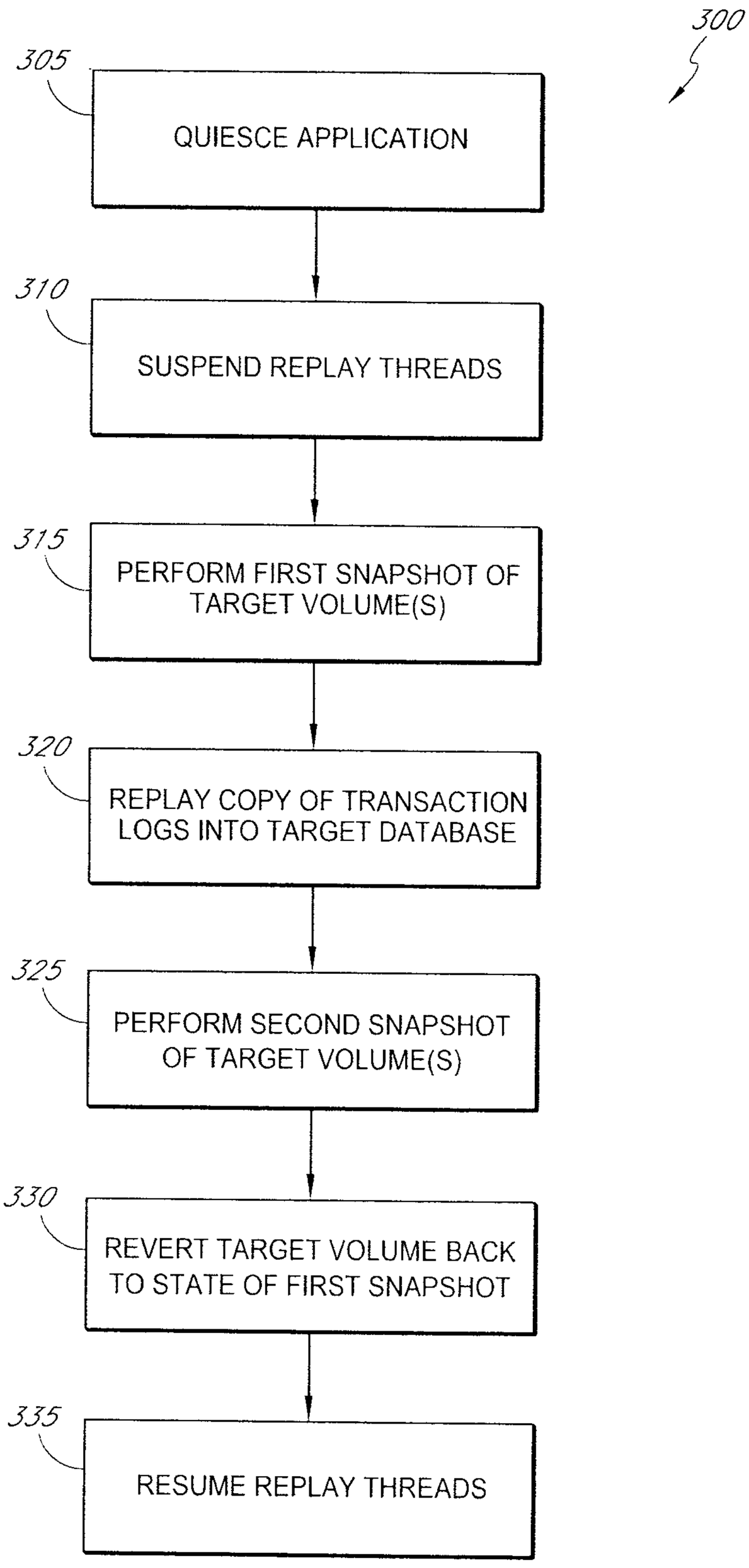

300
305
QUIESCE APPLICATION
310
SUSPEND REPLAY THREADS
315
PERFORM FIRST SNAPSHOT OF TARGET VOLUME(S)
320
REPLAY COPY OF TRANSACTION LOGS INTO TARGET DATABASE
325
PERFORM SECOND SNAPSHOT OF TARGET VOLUME(S)
330
REVERT TARGET VOLUME BACK TO STATE OF FIRST SNAPSHOT
335
RESUME REPLAY THREADS

FIG. 3

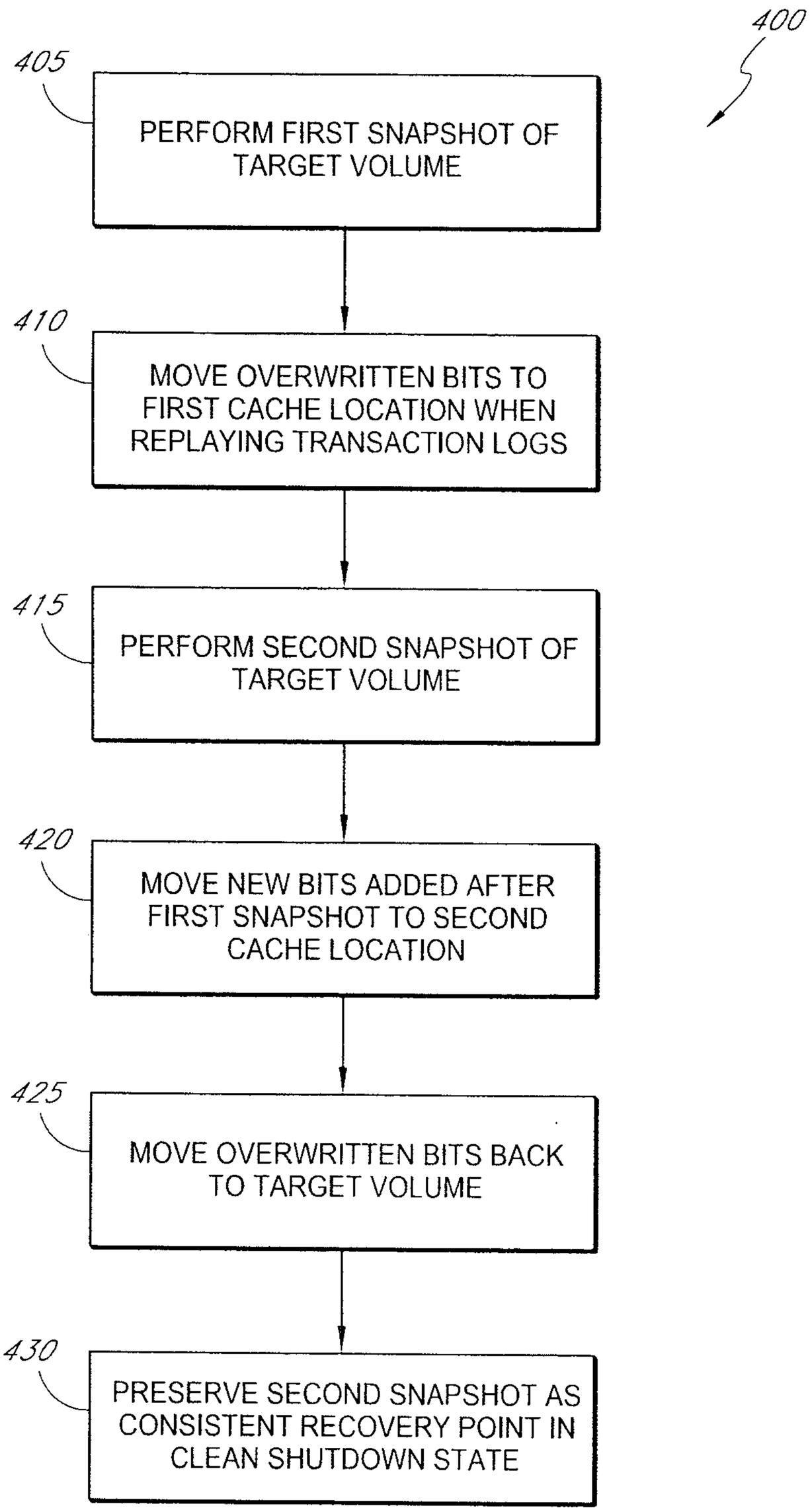

400
405
PERFORM FIRST SNAPSHOT OF TARGET VOLUME
410
MOVE OVERWRITTEN BITS TO FIRST CACHE LOCATION WHEN REPLAYING TRANSACTION LOGS
415
PERFORM SECOND SNAPSHOT OF TARGET VOLUME
420
MOVE NEW BITS ADDED AFTER FIRST SNAPSHOT TO SECOND CACHE LOCATION
425
MOVE OVERWRITTEN BITS BACK TO TARGET VOLUME
430
PRESERVE SECOND SNAPSHOT AS CONSISTENT RECOVERY POINT IN CLEAN SHUTDOWN STATE

FIG. 4

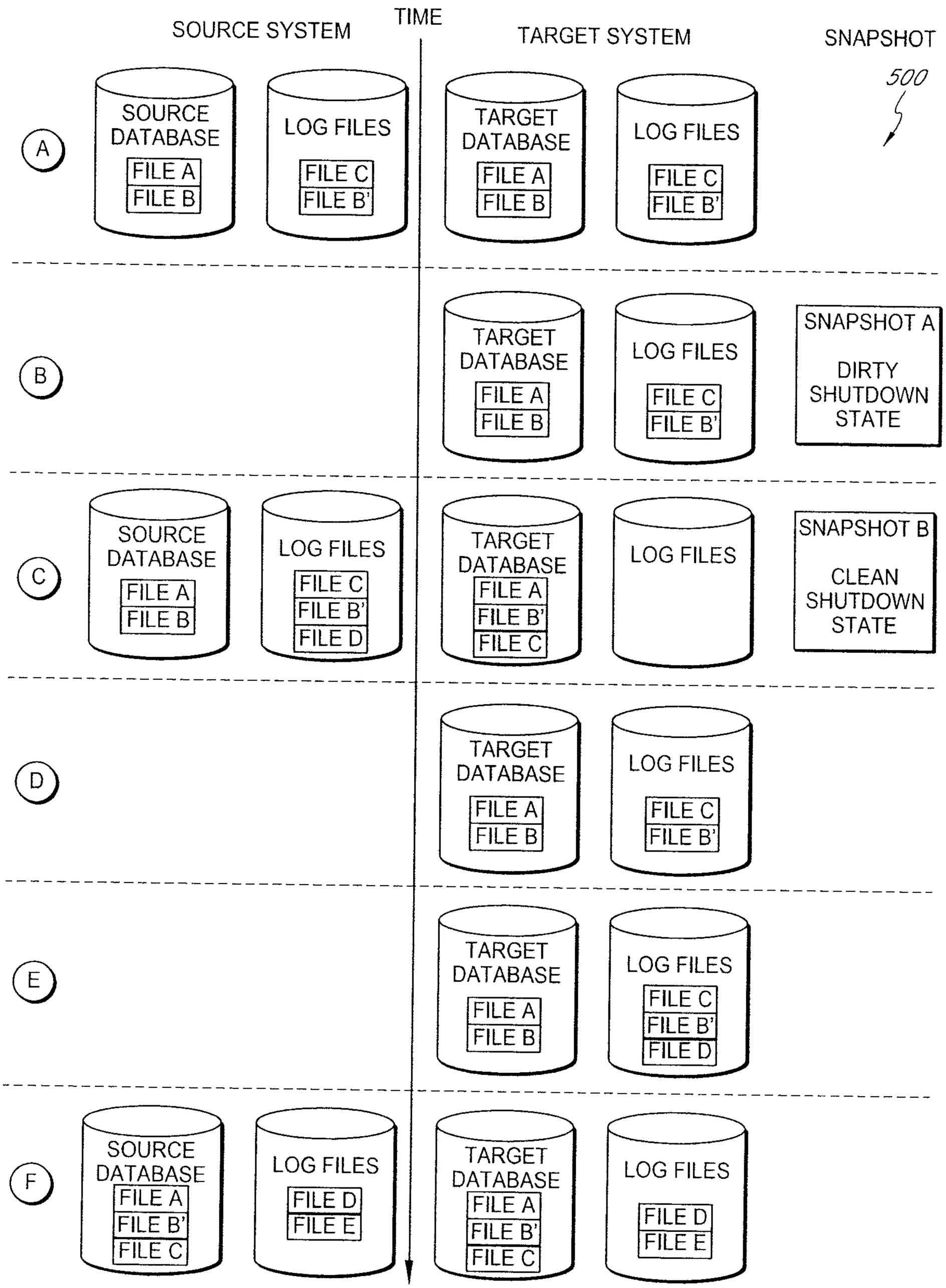

TIME
SOURCE SYSTEM
TARGET SYSTEM
SNAPSHOT
500
A
SOURCE DATABASE
FILE A
FILE B
LOG FILES
FILE C
FILE B'
TARGET DATABASE
FILE A
FILE B
LOG FILES
FILE C
FILE B'
B
TARGET DATABASE
FILE A
FILE B
LOG FILES
FILE C
FILE B'
SNAPSHOT A
DIRTY SHUTDOWN STATE
C
SOURCE DATABASE
FILE A
FILE B
LOG FILES
FILE C
FILE B'
FILE D
TARGET DATABASE
FILE A
FILE B'
FILE C
LOG FILES
SNAPSHOT B
CLEAN SHUTDOWN STATE
D
TARGET DATABASE
FILE A
FILE B
LOG FILES
FILE C
FILE B'
E
TARGET DATABASE
FILE A
FILE B
LOG FILES
FILE C
FILE B'
FILE D
F
SOURCE DATABASE
FILE A
FILE B'
FILE C
LOG FILES
FILE D
FILE E
TARGET DATABASE
FILE A
FILE B'
FILE C
LOG FILES
FILE D
FILE E

FIG. 5

TIME

SOURCE SYSTEM

TARGET SYSTEM

SNAPSHOT

500

A

SOURCE DATABASE
FILE A
FILE B

LOG FILES
FILE C
FILE B'

TARGET DATABASE
FILE A
FILE B

LOG FILES
FILE C
FILE B'

B

TARGET DATABASE
FILE A
FILE B

LOG FILES
FILE C
FILE B'

SNAPSHOT A

DIRTY SHUTDOWN STATE

C

SOURCE DATABASE
FILE A
FILE B

LOG FILES
FILE C
FILE B'
FILE D

TARGET DATABASE
FILE A
FILE B'
FILE C

LOG FILES

SNAPSHOT B

CLEAN SHUTDOWN STATE

D

TARGET DATABASE
FILE A
FILE B

LOG FILES
FILE C
FILE B'

E

TARGET DATABASE
FILE A
FILE B

LOG FILES
FILE C
FILE B'
FILE D

F

SOURCE DATABASE
FILE A
FILE B'
FILE C

LOG FILES
FILE D
FILE E

TARGET DATABASE
FILE A
FILE B'
FILE C

LOG FILES
FILE D
FILE E