RECONCILING A DISTRIBUTED DATABASE FROM HIERARCHICAL VIEWPOINTS

Inventor: Jason Lucas, Burlingame, CA (US)
Assignee: TAGGED, INC., San Francisco, CA (US)

Appl. No.: 13/564,147
Filed: Aug. 1, 2012

Related U.S. Application Data
Provisional application No. 61/513,932, filed on Aug. 1, 2011.

Publication Data
Publication No.: US 2013/0036105 A1
Publication Date: Feb. 7, 2013

Publication Classification
Int. Cl. G06F 17/30 (2006.01)
U.S. Cl. 707/703; 707/E17.014; 707/E17.005

ABSTRACT
Embodiments of the present disclosure include systems and methods for controlling concurrency of database transactions by hierarchically maintaining transaction consistently in a distributed database system from various viewpoints. Hierarchically maintaining the transaction consistency ensures the serializability of database transactions in the distributed database system and improves the overall performance (e.g., response time) from the perspective of clients of the distributed database system.
IDENTIFY A PLURALITY OF TRANSACTION SEQUENCES BASED ON A PLURALITY OF DATABASE QUERIES, WHEREIN EACH DATABASE QUERY INDICATES ONE OR MORE DATABASE TRANSACTIONS INITIATED BY AN APPLICATION RUNNING ON ONE OF A PLURALITY OF CLIENTS IN THE DISTRIBUTED DATABASE

SELECT A SUBSET OF THE PLURALITY OF TRANSACTION SEQUENCES

GENERATE AN INTERMEDIATE SHARED TRANSACTION SEQUENCE TO CONTINUOUSLY MAINTAIN TRANSACTION CONSISTENCY AMONG THE SUBSET OF THE PLURALITY OF TRANSACTION SEQUENCES

MAINTAIN THE INTERMEDIATE SHARED TRANSACTION SEQUENCE BY CONCURRENTLY AND ASYNCHRONOUSLY RECONCILING THE SUBSET OF THE PLURALITY OF TRANSACTION SEQUENCES INTO THE INTERMEDIATE SHARED TRANSACTION SEQUENCE

COMMIT THE DATABASE TRANSACTIONS IN THE INTERMEDIATE SHARED TRANSACTION SEQUENCE TO A GLOBAL TRANSACTION SEQUENCE THAT IS REPLICATED ACROSS THE DISTRIBUTED DATABASE

FIG. 4
RECEIVE QUERY

PROCESS QUERY TO IDENTIFY ASSERTIONS THAT REQUIRE CONSENSUS

QUERY PASSIVE LEARNERS FOR HISTORY OF ASSERTIONS

DOES CONSENSUS EXISTS?

FALL BACK CONSISTENTLY IN TIME ACROSS ALL ASSERTIONS FOR EACH TRANSACTION SEQUENCE UNTIL CONSENSUS ACHIEVED AMONG ASSERTIONS OR ALL ASSERTIONS TRAVERSED

WAS CONSENSUS ACHIEVED?

DOES SPECIFIC RECONCILIATION PROCEDURE EXIST?

TRANSACTION ABORTED

FIG. 8
RECONCILING A DISTRIBUTED DATABASE FROM HIERARCHICAL VIEWPOINTS

CLAIM OF PRIORITY

This application claims priority to U.S. Provisional Patent Application No. 61/513,932 entitled “Reconciling a Distributed Database from Hierarchical Viewpoints,” which was filed on Aug. 1, 2011, Attorney Docket No. 58520-8006. US01, the contents of which are expressly incorporated by reference herein.

CROSS-REFERENCE TO RELATED APPLICATION

This application is related to co-pending U.S. patent application Ser. No. ____, entitled “Generalized Reconciliation in a Distributed Database,” also by Jason Lucas, which was filed on Aug. 1, 2012, Attorney Docket No. 58520-8007.US01, the contents of which are expressly incorporated by reference herein.

This application is related to co-pending U.S. patent application Ser. No. ____, entitled “Systems and Methods for Asynchronous Distributed Database Management,” also by Jason Lucas, which was filed on Aug. 1, 2012, Attorney Docket No. 58520-8008.US01, the contents of which are expressly incorporated by reference herein.

TECHNICAL FIELD

Embodiments of the present disclosure generally relate to database management techniques and, more particularly to reconciling and/or otherwise maintaining a distributed database from various hierarchical viewpoints.

BACKGROUND

A distributed database is a database in which storage devices are not all attached to a common central processing unit (CPU). A distributed database may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers at multiple physical locations. The locations or sites of a distributed system may be spread over a large area (such as the United States or the world) or over a small area (such as a building or campus). The collections of data in the distributed database can also be distributed across multiple physical locations.

Typically, it is an object of a distributed database system to allow many users (clients or applications) use of the same information within the collection of data at the same time while making it seem as if each user has exclusive access to the entire collection of data. The distributed database system should provide this service with minimal loss of performance (latency) and maximal transaction throughput. That is, a user at location A must be able to access (and perhaps update) data at location B. If the user updates information, the updates must be propagated throughout the resources of the distributed database system to maintain consistency in the distributed database system.

A distributed database typically comes in one of two forms: synchronous or asynchronous. A synchronous database is a form of distributed database technology in which all data across the network is continuously kept up-to-date so that a user at any site can access data anywhere on the network at any time and get the same answer. Synchronous technology ensures data integrity and minimizes the complexity of knowing where the most recent copy of data is located. However, synchronous technology often results in very slow response times because the distributed database management system must spend considerable time checking that an update is accurately and completely propagated across the network.

A more common database is an asynchronous database. An asynchronous database is a form of distributed database technology in which copies of replicated data are kept at different nodes (or resources) so that local servers can access data without reaching out across the network. With asynchronous technology, there is usually some delay in propagating data updates across the remote databases, so some degree of at least temporary inconsistency is tolerated. Asynchronous technology tends to have better response times than synchronous technology because some of the updates can occur locally and data replicas can be synchronized in predetermined intervals across the network. However, synchronizing the replicas and serializing the database transactions to maintain concurrency can be an arduous task.

Additionally, with asynchronous technology, updates or database transactions must be serialized in the distributed database system to maintain consistency and/or concurrency. If transactions were executed in serial order, concurrency conflicts would never occur because each such transaction would be the only transaction executing on the system at a given time and would have exclusive use of the system’s resources. Any new transactions would see the results of previous transactions, plus its changes, and would not see the results of transactions that have yet to start. In operation, transactions typically execute concurrently and require simultaneous access and modification to the same resources. Thus, maintaining consistency in an asynchronous distributed database system can be very complex and also result in unacceptable response times.

Concurrency control schemes have been developed to control concurrency. Pessimistic concurrency control schemes control concurrency by preventing invalid use of resources. When one transaction attempts to use a resource in a way that could possibly invalidate the way another transaction has used the resource, pessimistic concurrency control schemes direct the requesting transaction to wait (e.g., a locking or restricted access scheme) until the resource is available for use without potential conflict. However, with pessimistic concurrency control schemes there needs to be mechanisms in place to detect deadlocks, or cycles of transactions all waiting for each other. Additionally, clients must often wait for resources unnecessarily.

Conversely, optimistic schemes control concurrency by detecting invalid use after the fact (e.g., by using resources and subsequently obtaining consensus). Optimistic concurrency control schemes optimize the case where conflict is rare. The basic idea is to divide a transaction’s lifetime into three phases: read, validate and publish. During the read phase, a transaction acquires resources without regard to conflict or validity, but it maintains a record of the set of resources it has used (a ReadSet) and the set of resources it has modified (a WriteSet). During the validation phase, the optimistic concurrency control scheme examines the ReadSet of the transaction and decides whether the current state of those resources has since changed. If the ReadSet has not changed, then the optimistic assumptions of the transaction are proved to have been right, and the system publishes the WriteSet, committing the transaction’s changes. If the ReadSet has changes, then the optimistic assumption of the transaction are
proved to be wrong, and the system aborts the transaction resulting in a loss of all changes. [0012] Unfortunately, optimistic concurrency schemes may also appear to be slow because they require consensus to be achieved among the resources prior to updating and/or committing changes to any number of database transactions in the distributed database system. Accordingly, optimistic concurrency control schemes also impede overall performance of the distributed database from the perspective of a user including system response times.

SUMMARY

[0013] Embodiments of the present disclosure include systems and methods for hierarchically maintaining and/or managing transaction consistency in distributed database systems from hierarchical viewpoints. In one embodiment, the systems and methods described herein teach selecting meaningful viewpoints for maintaining transaction consistency including performing intermediate reconciliation, if necessary, so the users’ perception of computer behavior and performance is optimized. For example, a data set corresponding to several users interacting in a game together (e.g., combined users) can be reconciled first among just those users interacting in the game together. The data set can subsequently be maintained and/or reconciled globally from the combined users’ perspective to a global transaction sequence.

[0014] In accordance with various embodiments, a database management system (DBMS) can hierarchically maintain transaction consistency in a distributed database by identifying a plurality of transaction sequences based on a plurality of database queries, wherein each database query indicates one or more database transactions initiated by an application running on one of a plurality of clients in the distributed database, selecting a subset of the plurality of transaction sequences, and generating an intermediate shared transaction sequence to continuously maintain transaction consistency among the subset of the plurality of transaction sequences, wherein intermediate shared transactions maintained in the intermediate shared transaction sequence are subsequently used to achieve global transaction consistency via a global transaction sequence that is replicated across a plurality of resources of the distributed database.

[0015] In one embodiment, the DBMS hierarchically maintains transaction consistency in a distributed database by replicating the global transaction sequence across the plurality of resources of the distributed database.

[0016] In one embodiment, the transaction sequence of the subset of the plurality of transaction sequences indicates a causal history of database transactions from the perspective of one of the applications.

[0017] In one embodiment, the DBMS hierarchically maintains transaction consistency in a distributed database by maintaining the intermediate shared transaction sequence, wherein maintaining the intermediate shared transaction sequence comprises asynchronously reconciling the subset of the plurality of transaction sequences into the intermediate shared transaction sequence. In one embodiment, each database transaction includes one or more assertions and reconciling the subset of the plurality of transaction sequences into the intermediate shared transaction sequence comprises determining the validity of each assertion.

[0018] In one embodiment, determining the validity of each assertion comprises moving consistently within each transaction sequence from a source transaction to a cause transaction until each assertion is validated.

[0019] In one embodiment, the intermediate shared transaction sequence represents a shared point of view as perceived from two or more applications operating on two or more clients of the plurality of clients.

[0020] In one embodiment, one or more transaction sequences of the subset of the plurality of transaction sequences are selected based on the applications that initiated the one or more transaction sequences.

[0021] In one embodiment, one or more transaction sequences of the subset of the plurality of transaction sequences are selected based on geographic locations of one or more clients associated with the one or more transaction sequences.

[0022] In one embodiment, one or more transaction sequences of the subset of the plurality of transaction sequences are selected based on an attribute of one or more of the plurality of clients associated with the one or more transaction sequences.

[0023] In one embodiment, the DBMS hierarchically maintains transaction consistency in a distributed database by committing the shared transactions in the intermediate shared transaction sequence to maintain the global transaction sequence, wherein committing the shared transactions to the global transaction sequence comprises reconciling one or more of the shared transactions in the intermediate shared transaction sequence with other database transactions in the distributed database.

[0024] In one embodiment, each database transaction includes one or more assertions and reconciling comprises achieving consensus among a plurality of database resources regarding the validity of each assertion. In one embodiment, achieving the consensus comprises moving consistently within each transaction sequence from a source database transaction to cause database transaction until each assertion is validated.

[0025] In one embodiment, the DBMS hierarchically maintains transaction consistency in a distributed database by prior to committing the shared transactions to the global transaction sequence, notifying one of the applications that the associated database query is completed.

[0026] In one embodiment, the DBMS hierarchically maintains transaction consistency in a distributed database by committing other uncommitted database transactions of the plurality of database transactions to the global transaction sequence, wherein the other uncommitted database transactions are not in the intermediate shared transaction sequence.

[0027] In one embodiment, the subset of the plurality of transaction sequences is selected based on a relation between the users of a first application.

[0028] In one embodiment, each user has a user profile associated with the first application and wherein the subset of the plurality of transactions sequences is selected based on a relation between the user profiles. In one embodiment, the subset of the plurality of transaction sequences is selected based on a type of the first application. In one embodiment, the first application comprises a multi-user online interactive game. In one embodiment, the subset of the plurality of transaction sequences is selected based on a social rank in the multi-user online interactive game.

[0029] In accordance with various embodiments, a DBMS can hierarchically maintain transaction consistency in a distributed database. The DBMS can include a processing unit,
an interface, and a memory unit. The interface can be configured to receive a plurality of database queries, wherein each database query indicates one or more database transactions initiated by an application running on one of a plurality of clients in a distributed database system. The memory unit can have instructions stored thereon, wherein the instructions, when executed by the processing unit, cause the processing unit to identify a plurality of transaction sequences based on the plurality of database queries, select a subset of the plurality of transaction sequences, and generate an intermediate shared transaction sequence to maintain transaction consistency among the subset of the plurality of transaction sequences.

In accordance with various embodiments, a DBMS can hierarchically maintain transaction consistency in a distributed database by receiving a plurality of database transactions from a plurality of client systems in the distributed database system, wherein each transaction sequence indicates uncommitted database transactions initiated by an application running on one of the plurality of client systems. The DBMS can identify a plurality of transaction sequences based on the plurality of database transactions, wherein each database transaction is initiated by an application running on one of a plurality of clients in the distributed database system. The DBMS can select a subset of the plurality of transaction sequences based on a first criteria. The DBMS can generate an intermediate shared transaction sequence to maintain transaction consistency among the subset of the plurality of transaction sequences. The DBMS can commit the database transactions indicated by the intermediate shared transaction sequence to a global transaction sequence, but prior to committing the database transactions indicated by the intermediate shared transaction sequence to the global transaction sequence, the DBMS can send a notification indicating a commit or failure of one or more of the database transactions in the intermediate shared transaction sequence to the application on client device that initiated the database request.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 depicts a block diagram of an example distributed database environment illustrating a plurality of distributed database sites and client systems within which various features of the present invention may be utilized, according to one embodiment.

FIG. 2 depicts a block diagram of an example node in a distributed database environment within which various features of the present invention may be utilized, according to an embodiment.

FIG. 3 depicts a block diagram of the components of a database management system for hierarchically maintaining transaction consistency in a distributed database system, according to an embodiment.

FIG. 4 depicts a flow diagram illustrating an example process for hierarchically maintaining transaction consistency in a distributed database system, according to an embodiment.

FIGS. 5A and 5B depict diagrams illustrating an example of an intermediate reconciliation process in a distributed database system, according to an embodiment.

FIGS. 6A and 6B depict transaction sequences illustrating an example intermediate reconciliation process in a distributed database system, according to an embodiment.

FIG. 7 depicts a diagram illustrating an example of an intermediate reconciliation process in a distributed database system, according to an embodiment.

FIG. 8 depicts a flow diagram illustrating an example process for hierarchically maintaining transaction consistency in a distributed database system, according to an embodiment.

FIG. 9 shows a diagrammatic representation of a machine in the example form of a computer system within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed, according to an embodiment.

DETAILED DESCRIPTION

The systems and methods disclosed provide for controlling concurrency of database transactions by hierarchically maintaining transaction consistency in a distributed database system from various viewpoints. Hierarchically maintaining the transaction consistency ensures the serializability of database transactions in the distributed database system and improves the overall performance (e.g., response time) from the perspective of clients of the distributed database system.

The distributed database systems described herein can be comprised of a number of resources or nodes. In some embodiments, each of the resources or nodes has a system clock. Prior art mechanisms typically use the clocks and locking based mechanisms to control interleaving of operations or database transactions from the resources. However, the distributed database resources described herein do not rely on their system clocks in order to serialize the order of requests. Rather, it is an object of the current disclosure to increase concurrency by interleaving database transactions based on the underlying assertions upon which the database transactions rely. As described herein, each assumption is controlled with assertions that can be used in lieu of locks to permit interleaving of operations and increased concurrency.

The interleaving (or reconciling) of the database transactions at various hierarchical viewpoints increases response time as perceived from users of the distributed database system.

The following description and drawings are illustrative and are not to be construed as limiting. Numerous specific details are described to provide a thorough understanding of the disclosure. However, in certain instances, well-known or conventional details are not described in order to avoid obscuring the description. References to one or an embodiment in the present disclosure can be, but not necessarily are, references to the same embodiment; and, such references mean at least one of the embodiments.

Reference in this specification to “one embodiment” or “an embodiment” means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the disclosure. The appearances of the phrase “in one embodiment” in various places in the specification are not necessarily all referring to the same embodiment, nor are separate or alternative embodiments mutually exclusive of other embodiments. Moreover, various features are described which may be exhibited by some embodiments and not by others. Similarly, various requirements are described which may be requirements for some embodiments but not other embodiments.

The terms used in this specification generally have their ordinary meanings in the art, within the context of the disclosure, and in the specific context where each term is
used. Certain terms that are used to describe the disclosure are discussed below, or elsewhere in the specification, to provide additional guidance to the practitioner regarding the description of the disclosure. For convenience, certain terms may be highlighted, for example using italics and/or quotation marks. The use of highlighting has no influence on the scope and meaning of a term; the scope and meaning of a term is the same, in the same context, whether or not it is highlighted. It will be appreciated that the same thing can be said in more than one way.

[0045] Consequently, alternative language and synonyms may be used for any one or more of the terms discussed herein, nor is any special significance to be placed upon whether or not a term is elaborated or discussed herein. Synonyms for certain terms are provided. A recital of one or more synonyms does not exclude the use of other synonyms. The use of examples anywhere in this specification including examples of any terms discussed herein is illustrative only, and is not intended to further limit the scope and meaning of the disclosure or of any exemplified term. Likewise, the disclosure is not limited to various embodiments given in this specification.

[0046] Without intent to further limit the scope of the disclosure, examples of instruments, apparatus, methods and their related results according to the embodiments of the present disclosure are given below. Note that titles or subtitles may be used in the examples for convenience of a reader, which in no way should limit the scope of the disclosure. Unless otherwise defined, all technical and scientific terms used herein have the same meaning as commonly understood by one of ordinary skill in the art to which this disclosure pertains. In the case of conflict, the present document, including definitions will control.

[0047] Embodiments of the present disclosure include various steps, which will be described below. The steps may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps may be performed by a combination of hardware, software and/or firmware.

[0048] Embodiments of the present disclosure may be provided as a computer program product, which may include a machine-readable medium having stored thereon instructions, which may be used to program a computer (or other electronic devices) to perform a process. The machine-readable medium may include, but is not limited to, floppy disks, optical disks, compact disc read-only memories (CD-ROMs), and magneto-optical disks, ROMs, random access memories (RAMs), erasable programmable read-only memories (EPROMs), electrically erasable programmable read-only memories (EEPROMs), field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), vehicle identity modules (VIMs), magnetic or optical cards, flash memory, or other type of media/machine-readable medium suitable for storing electronic instructions.

[0049] Moreover, embodiments of the present invention may also be downloaded as a computer program product or data to be used by a computer program product, wherein the program, data, and/or instructions may be transferred from a remote computer or mobile device to a requesting computer or mobile device by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection). In some cases, parts of the program, data, or instructions may be provided by external networks such as the telephony network (e.g., Public Switched Telephony Network, cellular, Wi-Fi, and other voice, data, and wireless networks) or the Internet. The communications link may be comprised of multiple networks, even multiple heterogeneous networks, such as one or more border networks, voice networks, broadband networks, service provider networks, Internet Service Provider (ISP) networks, and/or Public Switched Telephone Networks (PSTNs), interconnected via gateways operable to facilitate communications between and among the various networks.

Terminology

[0050] Brief definitions of terms used throughout this application are given below.

[0051] The terms “connected” or “coupled” and related terms are used in an operational sense and are not necessarily limited to a direct connection or coupling.

[0052] The term “embodiments,” phrases such as “in some embodiments,” “in various embodiments,” and the like, generally mean the particular feature(s), structure(s), method(s), or characteristic(s) following or preceding the term or phrase is included in at least one embodiment of the present invention, and may be included in more than one embodiment of the present invention. In addition, such terms or phrases do not necessarily refer to the same embodiments.

[0053] If the specification states a component or feature “may”, “can”, “could”, or “might” be included or have a characteristic, that particular component or feature is not required to be included or have the characteristic.

[0054] The term “module” refers broadly to a software, hardware, and/or firmware (or any combination thereof) component. Modules are typically functional components that can generate useful data or other output using specified input(s). A module may or may not be self-contained. An application program (also called an “application”) may include one or more modules, and/or a module can include one or more application programs.

[0055] The term “responsive” includes completely and partially responsive.

Example Distributed Database Environment

[0056] An example of a distributed database environment 100, representing a plurality of distributed database sites and client systems, within which various features of the present invention may be utilized, will now be described with reference to FIG. 1. In this example, the distributed database environment 100 comprises a plurality of nodes 10, a plurality of client systems 25, and a network 150. Each node 10 may be located at a different site or geographic location. Similarly, each client system 25 may be located anywhere within connectivity of network 150.

[0057] In this example, the nodes 10 are in communication with other nodes 10 via network 150. The nodes 10 may be centralized database systems such as data warehouses or data marts, remote sites such as desktop personal computers, portable computers or other mobile computing devices, or any other type of data processors. As shown in this example, the nodes 10 include database management systems 18 in communication with distributed databases 20. The database management systems 18 may be in communication with a database 20 via any communication means for communicating data and/or control information. Although not shown for sim-
plicity, database management system 18 may also include both a distributed database management system and a local database management system. Similarly, although not shown, database 20 may include both a distributed database and a local database.

[0058] In some embodiments, one or more of the distributed database management systems 18 may be designated the master management system or host server system. The master management system may, in some cases, be responsible for reconciling database transactions and/or database transaction sequences as disclosed herein; although alternative configurations are possible.

[0059] The network 150, over which client systems 25 and nodes 10 communicate, may be a local area network, a metropolitan area network, a wide area network, a global data communications network such as the Internet, a private “intranet” or “extranet” network or any other suitable data communication medium—including combinations or variations thereof. For example, the Internet can provide file transfer, remote login, email, news, RSS, and other services through any known or convenient protocol, such as, but is not limited to the TCP/IP protocol, Open System Interconnections (OSI), FTP, UPnP, iSCSI, NSF, ISDN, PDH, RS-232, SDH, SONET, etc.

[0060] Alternatively or additionally, the network 150 can be any collection of distinct networks operating wholly or partially in conjunction to provide connectivity to the client systems 25 and nodes 10 and may appear as one or more networks to the serviced systems and devices. In one embodiment, communications to and from client systems 25 can be achieved by, an open network, such as the Internet, or a private network, such as an intranet and/or the extranet. In one embodiment, communications can be achieved by a secure communications protocol, such as secure sockets layer (SSL), or transport layer security (TLS).

[0061] In addition, communications can be achieved via one or more wireless networks, such as, but is not limited to, one or more of a Local Area Network (LAN), Wireless Local Area Network (WLAN), a Personal area network (PAN), a Campus area network (CAN), a Metropolitan area network (MAN), a Wide area network (WAN), a Wireless wide area network (WWAN), Global System for Mobile Communications (GSM), Personal Communications Service (PCS), Digital Advanced Mobile Phone Service (D-Amps), Bluetooth, Wi-Fi, Fixed Wireless Data, 2G, 2.5G, 3G networks, enhanced data rates for GSM evolution (EDGE), General packet radio service (GPRS), enhanced GPRS, messaging protocols such as TCP/IP, SMS, MMS, extensible messaging and presence protocol (XMPP), real-time messaging protocol (RTMP), instant messaging and presence protocol (IMPP), instant messaging, USSD, IRC, or any other wireless data networks or messaging protocols.

[0062] The client systems (or clients) 25 are in communication with one or more nodes 10 via network 150. Client systems 25 can be any system and/or device, and/or any combination of devices/systems that is able to establish a connection with another device, a server and/or other systems. The client systems 25 typically include display or other output functionalities to present data exchanged between the devices to a user. For example, the client systems 25 can be, but are not limited to, a server desktop, a laptop computer, a computer cluster, a mobile computing device such as a notebook, a laptop computer, a handheld computer, a mobile phone, a smart phone, a PDA, a Blackberry device, a Treo, and/or an iPhone, etc. In one embodiment, client systems 25 are coupled to the network 150. In some embodiments, the client systems 25 may be directly connected to one another or to nodes 10.

[0063] The client systems 25 include a query interface 22 and/or one or more applications 26. An application 26 may execute on client 25 and may include functionality for invoking query interface 22 for transferring a database query to a database server for processing. The application 26 may invoke the query interface 22 for reading data from or writing data to a database table of a distributed database 20. In general, application 26 and query interface 22 may be any type of interpreted or executable software code such as a kernel component, an application program, a script, a linked library, or any object with method, including combinations or variations thereof. In one example, the application 26 comprises a multi-user interactive game; however, it is appreciated that other applications are also possible.

[0064] In some embodiments, one or more of the database management systems 18 maintain one or more transaction sequences for each client system 25 by asynchronously and concurrently reconciling the database transactions from various hierarchical viewpoints. The transaction sequences can comprise one or more database transactions. In operation, the database transactions may be generated by an application 26 within client system 25 and transferred to the associated database management system 18 via a query generated by query interface 22. As shown in the example of FIG. 1, the query is transferred over network 150 and received at one of the database management systems 18.

[0065] In some embodiments, each transaction sequence may be a continuous independent sequence or a linear time model that indicates database transactions from a personal point of view. The personal point of view may be, for example, the point of view of one or more applications running on a client and/or the point of view of a client system or an operator (e.g., user or player) of the client system. A shared point of view or shared transaction sequence may be the point of view of as perceived from two or more applications running on two or more clients (or client systems or operators).

[0066] In some embodiments, the transaction sequences may be represented by a graph such as a causality graph or a serialization graph. Causality graphs and serialization graphs contain information about current and historic database transactions or operations, such as database queries received from a client system.

[0067] In some embodiments, the database management system 18 maintains the associated transaction sequences for the client systems 25 and asynchronously and concurrently reconciles the database transactions within the transaction sequences with other relevant database transactions in other transaction sequences received within the distributed database system.

[0068] In some embodiments, each database transaction operates with a set of assumptions upon which the database transaction relies. As described herein, the assumptions are controlled with assertions that can be used in lieu of locks to permit interleaving of operations and increased concurrency. In some embodiments, the assertions enforce consistency using various mechanisms such as, for example, a multi-version concurrency control (MVCC) mechanism. As described herein, the concurrency control mechanisms facilitate the ability to seek a time in the past during which asser-
tions are true. This process is referred to herein as “time traveling,” and is discussed in greater detail with reference to FIG. 8.

[0069] In some embodiments, database 20 includes a global transaction sequence containing the committed database transactions. In some embodiments, the global transaction sequence is replicated across some or all of the databases 20 in the distributed database environment 100.

[0070] FIG. 2 depicts a block diagram of an example node 210 in a distributed database environment 200, according to an embodiment. The distributed database environment 200 may be similar to the distributed database environment 100 of FIG. 1, although alternative configurations are possible.

[0071] In this example, node 210 includes a database management system 218 in communication with databases 220-D and 220-L (distributed and local, respectively), and a network 250. The network 250 may be any network such as, for example, network 150 of FIG. 1. The node 210 may be similar to the nodes 10 of FIG. 1; although alternative configurations are possible. In some embodiments, while each node includes a local database management system 219-L, only one master distributed database system 219-D exists. In this case, the distributed database system 219-D controls the interaction across database.

[0072] The database management system 218 further includes a distributed database management system 219-D, a local database management system 219-L, optional application programs 219-A. The distributed database management system 219-D coordinates access to the data at the various nodes. The distributed database management system 219-D may perform some or all of the follow functions:

[0073] 1. Keep track of where data is located in a distributed data dictionary. This includes presenting one logical database and schema to developers and users.
[0074] 2. Determine the location from which to retrieve requested data and the location at which to process each part of a distributed query without any special actions by the developer or user.
[0075] 3. If necessary, translate the request at one node using a local DBMS into the proper request to another node using a different DBMS and data model and return data to the requesting node in the format accepted by that node.
[0076] 4. Provide data management functions such as security, concurrency and deadlock control, global query optimization, and automatic failure recording and recovery.
[0077] 5. Provide consistency among copies of data across the remote sites (e.g., by using multiphase commit protocols).
[0078] 6. Present a single logical database that is physically distributed. One ramification of this view of data is global primary key control, meaning that data about the same business object are associated with the same primary key no matter where in the distributed database the data are stored, and different objects are associated with different primary keys.
[0079] 7. Be scalable. Scalability is the ability to grow, reduce in size, and become more heterogeneous as the needs of the business change. Thus, a distributed database must be dynamic and be able to change within reasonable limits without having to be redesigned. Scalability also means that there are easy ways for new sites to be added (or to subscribe) and to be initialized (e.g., with replicated data).

[0080] 8. Replicate both data and stored procedures across the nodes of the distributed database. The need to distribute stored procedures is motivated by the same reasons for distributing data.

[0081] 9. Transparently use residual computing power to improve the performance of database processing. This means, for example, the same database query may be processed at different sites and in different ways when submitted at different times, depending on the particular workload across the distributed database at the time of query submission.

[0082] 10. Permit different nodes to run different DBMSs. Middleware (see Chapter 9) can be used by the distributed DBMS and each local DBMS to mask the differences in query languages and nuances of local data.

[0083] 11. Allow different versions of application code to reside on different nodes of the distributed database. In a large organization with multiple, distributed servers, it may not be practical to have each server/node running the same version of software.

[0084] In one embodiment, each node includes both a local database system 219-L and a distributed database management system 219-D. In the example of FIG. 2, each site has a local DBMS 219-L that manages the local database 220-L stored at that site and a copy of the distributed DBMS database 220-D and the associated distributed data dictionary/directory (DD/DD). The distributed DD/DD contains the location of all data in the network, as well as data definitions.

[0085] Requests for data by users or application programs are first processed by the distributed DBMS 219-D, which determines whether the request is local or global. A local transaction is one in which the required data are stored entirely at the local site. A global transaction requires reference to data at one or more non-local sites to satisfy the request. For local transactions, the distributed DBMS 219-D passes the request to the local DBMS 219-L. For global transactions, the distributed DBMS 219-D routes the request to other sites as necessary. The distributed DBMSs at the participating sites exchange messages as needed to coordinate the processing of the transaction until it is completed (or aborted, if necessary).

[0086] FIG. 3 depicts a block diagram of the components of a database management system 350 for maintaining transaction consistency in a distributed database system from hierarchical viewpoints, according to an embodiment. The database management system 350 may be the database management system 18 of FIG. 1, although alternative configurations are possible.

[0087] The database management system 350 includes a network interface 302, a communications module 305, a database transaction reception module 310, a database transaction history module 315, a causality graph generation module 320, an assertion identification/extraction module 325, a shared transaction sequence module 330, and a global transactions sequence module 340. In an embodiment, the database management system 350 is also coupled to a database 345. The database 345 can be the database 20 of FIG. 1, although alternative configurations are possible. Additional or less modules can be included without deviating from the novel art of this disclosure. Furthermore, each module in the example of FIG. 3 can include any number and/or combination of sub-modules and/or systems, implemented with any combination of hardware and/or software.

[0088] The database management system 350, although illustrated as comprised of distributed components (physi-
cally distributed and/or functionally distributed), could be implemented as a collective element. In some embodiments, some or all of the modules, and/or the functions represented by each of the modules can be combined in any convenient or known manner. Furthermore, the functions represented by the modules can be implemented individually or in any combination thereof, partially or wholly, in hardware, software, or a combination of hardware and software.

[0089] In the example of FIG. 3, the network interface 302 can be a networking device that enables the database management system 350 to modulate data in a network with an entity that is external to the database management system 350, through any known and/or convenient communications protocol supported by the host and the external entity. The database management system 350 can include one or more of a network adaptor card, a wireless network interface card, a router, an access point, a wireless router, a switch, a multilayer switch, a protocol converter, a gateway, a bridge, bridge router, a hub, a digital media receiver, and/or a repeater.

[0090] One embodiment of the database management system 350 includes the communications module 305. The communications module 305 can be any combination of software agents and/or hardware modules able to identify, detect, track, manage, receive, record, and/or process data access requests. The communications module 305, when in operation, is able to communicate with the network interface 302 to identify, detect, track, manage, receive, record, and/or process data access requests including, but not limited to, database queries and/or database transactions from client systems and/or other nodes in the distributed database system.

[0091] One embodiment of the database management system 350 includes the database transaction reception module 310. The database transaction reception module 310 can be any combination of software agents and/or hardware components able to receive and process data requests from the client devices and other nodes. For example, the database transaction reception module 310 is configured to receive and process database queries from the client devices and other data request from other nodes in the system. The database transaction reception module 310 may then segment, route, and/or otherwise process the requests and/or identify the database transactions with the data requests or queries.

[0092] One embodiment of the database management system 350 includes the database transaction history module 315. The database transaction history module 315 can be any combination of software agents and/or hardware components able to track and store historical transactions. For example, the history may include transaction order, assumptions/assertions relied upon, etc. Advantageously, schemas do not need to include histories because the database keeps track of this information.

[0093] One embodiment of the database management system 350 includes the causality graph generation module 320. The causality graph generation module 320 can be any combination of software agents and/or hardware components able to interact with the transaction history module 315 to generate a causality graph for one or more database transactions in or indicated by a transaction sequence. For example, the causality graph generation module 320 can identify transaction sequences based on the received database queries. As discussed, the database queries indicate one or more database transactions. The causality graph generation module 320 can use the database transaction information to interact with the database transaction history module 315 in order to identify the historical transactions upon which the current database transaction relies and build a causality graph based on this history information.

[0094] In one embodiment, the causality graph generation module 320 generates a causality graph indicating the one or more assertions upon which each database transaction relies. For example, in some embodiments, concurrency control schemes control concurrency by detecting invalid use after the fact. These concurrency controls may divide a transaction's existence into read, validate, and publish phases. During the read phase, the scheme acquires assumptions from one or more distributed database resources regarding the underlying values of the assumptions upon which the transaction relies without regard to conflict or validity of these assumptions. The transaction sequences themselves and/or the database transaction history module may indicate the set of resources and/or assumptions relied upon for each database transaction in transaction sequence. In some embodiments, assertions may be, for example, database key values; although alternative configurations are possible.

[0095] One embodiment of the database management system 350 includes the assertion identification/extractor module 325. The assertion identification/extractor module 325 can be any combination of software agents and/or hardware components able to identify and/or extract the assertions associated with one or more database transactions. For example, the assertion identification/extractor module 325 may process database transactions, transaction sequences, and/or database queries to identify and/or extract the underlying assertions upon which the database transactions rely.

[0096] In one embodiment, each database transaction operates with a set of assumptions on which the database transaction relies. As described herein, the assumptions are controlled with assertions that can be used in lieu of locks to permit interleaving of operations and increased concurrency. The assertions can enforce consistency using various mechanisms such as, for example, a multi-version concurrency control (MVCC) mechanism as described herein.

[0097] One embodiment of the database management system 350 includes the shared transaction sequence module 330. The shared transaction sequence module 330 can be any combination of software agents and/or hardware components able to maintain transaction consistency among a subset of a plurality of transaction sequences received by the database management system. In this example, the shared transaction sequence module 330 includes a selection engine 332, a generation engine 334, a consensus engine 336, and a reconciliation engine 338.

[0098] In one embodiment the selection engine 332 is configured to select a subset of a plurality of transaction sequences for which to generate an intermediate shared transaction sequence to continuously and asynchronously maintain transaction consistency. The selection engine 332 may select the subset of the plurality of transaction sequences based on any number of factors. For example, one or more transaction sequences of the subset of the plurality of transaction sequences can be selected based on the applications that initiated the one or more transaction sequences. Similarly, one or more transaction sequences of the subset of the plurality of transaction sequences can be selected based on geographic locations of one or more clients associated with the one or more transaction sequences. Likewise, one or more transaction sequences of the subset of the plurality of trans-
action sequences can be selected based on an attribute of one or more of the plurality of clients associated with the one or more transaction sequences.

Alternatively or additionally, one or more transaction sequences of the subset of the plurality of transaction sequences can be selected based on a relation or association between the users of the application or the application itself. For example, each user can have a user profile associated with an application running on a client system. The subset of the plurality of transaction sequences can be selected based on a relation between the user profiles. Additionally, in some embodiments, the relation between the profiles can be based on interactions between the users profiles via the applications. For example, if two players of an online interactive game are engaged in an alliance and a third user is not engaged in the alliance then the transaction sequences associated with those two users in the alliance can be selected.

Alternatively or additionally, one or more transaction sequences of the subset of the plurality of transaction sequences can be selected based on a relation or association between the applications themselves. That is the one or more transaction sequences can be selected based on the types of applications the users or players are using. For example, if two users are engaged in an online interactive game and the third user is not engaged in the same online interactive game or the third user is engaged in a different interactive game, then the transaction sequences associated with the two users engaged in the interactive game may be selected for the subset of the plurality of transaction sequences. Alternatively or additionally, one or more transaction sequences of the subset of the plurality of transaction sequences can be selected based on social rank in a multi-user online interactive game.

In one embodiment the generation engine 334 is configured to generate the shared transaction sequence in order to continuously and asynchronously maintain transaction consistency.

In one embodiment the consensus engine 336 is configured to achieve consensus among a plurality of database resources regarding the validity of each assertion. For example, the systems and methods described herein can operate according to the CAP theorem, also known as Brewer’s theorem. The CAP theorem states that it is impossible for a distributed computer system to simultaneously guarantee consistency, availability, and partition tolerance.

Consistency guarantees that all nodes of the distributed database see the same data at the same time. Availability guarantees that every request receives a response about whether the request was successful or failed. Partition tolerance guarantees that the system continues to operate despite arbitrary message loss. According to the CAP theorem, a distributed system can satisfy any two of the above guarantees at the same time, but not all three.

There are certain limitations on database system that maintain a distributed scalable state due, at least in part, to unreliable processors. One solution is allowing consensus. Consensus is the process of agreeing on a single result among a group of participants (or resources). Consensus protocols are the basis for the state machine approach to distributed computing. The state machine approach is a technique for converting an algorithm into a fault-tolerant, distributed implementation. Every potential fault must have a way to be dealt with and ad-hoc techniques often leave important cases of failures unresolved.

In some embodiments, the systems and methods described herein use consensus protocols such as, for example, the Paxos algorithm. The Paxos algorithm describes protocols for solving consensus in a network of unreliable processors. This problem becomes difficult when the participants or their communication medium experience failures. The Paxos approach provides a technique to ensure that all cases are handled safely. However, these cases may still need to be individually coded.

The Paxos protocols define a number of roles and describes the actions of the processes by their roles in the protocol: client, acceptor, proposer, learner, and leader. In typical implementations, a single processor may play one or more roles at the same time. This does not affect the correctness of the protocol—it is usual to coalesce roles to improve the latency and/or number of messages in the protocol.

The Paxos protocols include a spectrum of tradeoffs between the number of processors, number of message delays before learning the agreed value, the activity level of individual participants, number of messages sent, and types of failures. However, no fault-tolerant consensus protocol can guarantee progress.

Clients: Clients issue requests to the distributed system, and wait for a response. For instance, a write request on a file in a distributed file server. Acceptors: Acceptors act as the fault-tolerant “memory” of the protocol. Acceptors are collected into groups called Quorums. Any message sent to an Accepter must be sent to a Quorum of Accepters, and any message received from an Accepter is ignored unless a copy is received from each Accepter in a Quorum. Proposers: Proposers advocate a client request, attempting to convince the Accepters to agree on it, and acting as a coordinator to move the protocol forward when conflicts occur. Learners: Learners act as the replication factor for the protocol. Once a Client request has been agreed on by the Accepters, the Learner may take action (i.e., execute the request and send a response to the client). To improve availability of processing, additional Learners can be added. Leaders: Leaders are distinguished Proposers that are required to make progress. Many processes may believe they are leaders, but the protocol only guarantees progress if one of them is eventually chosen. If two processes believe they are leaders, it is possible to stall the protocol by continuously proposing conflicting updates. The safety properties are preserved regardless.

In one embodiment, the reconciliation engine 338 is configured to maintain the generated intermediate shared transaction sequence by continuously and asynchronously reconciling the subset of the plurality of transaction sequences into the intermediate shared transaction sequence. For example, the reconciliation engine 338 may reconcile database transactions included within the selected plurality of database transaction sequences before reconciling and/or otherwise committing the database transactions to a global transaction sequence. The database transactions can be reconciled according to the underlying assertions. That is, the assertions can be used in lieu of locks to permit interleaving of database transactions and increase concurrency.

One embodiment of the database management system 350 includes the global transaction sequence module 340. The global transaction sequence module 340 can be any combination of software agents and or hardware components able to maintain, reconcile, and commit database transactions to a global transaction sequence. The global transaction sequence module 340 may maintain, reconcile, and commit
database transactions from one or more of the generated shared sequences and/or from individual transaction sequences (e.g., from private sequences). In this example, the global transaction sequence module 340 includes a consensus engine 342, a reconciliation engine 344, and a commit engine 346.

[0111] In one embodiment, the consensus engine 342 is configured to achieve consensus among a plurality of systems (e.g., node or resources) of the distributed database system. The consensus engine 342 is similar to the consensus engine 336; however, the consensus engine 342 operates to achieve consensus among all of the plurality of database transactions both included within the subset of the plurality of selected database transaction sequences and not included within those sequences.

[0112] In one embodiment, the reconciliation engine 344 is configured to reconcile the plurality of database transactions. The reconciliation engine 344 can reconcile database transactions both included within the subset of the plurality of selected database transaction sequences and not included within those sequences.

[0113] In one embodiment, the commit engine 346 is configured to commit database transactions to the global transaction sequence. Advantageously, the database transactions from one or more intermediate shared transaction sequences already appear to have been committed to the users, although these transactions may not actually be committed until the commit engine 346 performs the common operation. To ensure data integrity for real-time, distributed update operations, the cooperating transaction managers can execute a commit protocol. The commit protocol is a well-defined procedure (involving an exchange of messages) to ensure that a global transaction is either successfully completed at each site or else aborted.

[0114] The most widely used protocol is called a two-phase commit. A two-phase commit protocol ensures that concurrent transactions at multiple sites are processed as though they were executed in the same, serial order at all sites. A two-phase commit works in two phases. To begin, the site originating the global transaction or an overall coordinating site sends a request to each of the sites that will process some portion of the transaction. Each site processes the subtransaction (if possible), but does not immediately commit (or store) the result to the local database. Instead, the result is stored in a temporary file. Additionally, each site locks (or prohibits others from updating) its portion of the database being updated and notifies the originating site when it has completed its subtransaction. When all sites have responded, the originating site now initiates the two-phase commit protocol.

[0115] In a prepare phase, a message is broadcast to every participating site (or node), asking whether that site is willing to commit its portion of the transaction at that site. Each site returns an "OK" or "not OK" message. An "OK" indicates that the remote site promises to allow the initiating request to govern the transaction at the remote database. Next, in a commit phase, the originating site collects the messages from all sites. If all are "OK," it broadcasts a message to all sites to commit the portion of the database handled at each site. However, if one or more responses are "not OK," it broadcasts a message to all sites to abort the transaction.

[0116] A limbo transaction can be identified by a timeout or polling. With a timeout (no confirmation of commit for a specified time period), it is not possible to distinguish between a busy or failed site. Polling can be expensive in terms of network load and processing time. With a two-phase commit strategy for synchronizing distributed data, committing a transaction is slower than if the originating location were able to work alone.

[0117] One embodiment of the database management system 350 includes the database 345. The database 345 can store any data items/entries including, but not limited to, software, descriptive data, images, system information, drivers, and/or any other data item utilized by the database management system and/or any other systems for operation. The database 345 may be coupled to the database management system 350. The database 345 may be managed by a database management system (DBMS), for example but not limited to, Oracle, DB2, Microsoft Access, Microsoft SQL Server, PostgreSQL, MySQL, FileMaker, etc. The user data repository 128 can be implemented via object-oriented technology and/or via text files, and can be managed by a distributed database management system, an object-oriented database management system (OODBMS) (e.g., ConceptBase, FastDB Main Memory Database Management System, JDOInstruments, ObjectDB, etc.), an object-relational database management system (ORDBMS) (e.g., Informix, Openink Virtuoso, VMDS, etc.), a file system, and/or any other convenient or known database management package.

[0118] FIG. 4 depicts a flow diagram illustrating an example process 400 for hierarchically maintaining transaction consistency in a distributed database, according to an embodiment. One or more database management systems, such as, for example, the database management systems 18 of FIG. 1, among other functions, maintain and/or reconcile the transaction consistency in the distributed database system from hierarchical viewpoints.

[0119] To begin, database queries are received by a database management system in the distributed database system. In operation, the database queries can be received by any number of database management systems in the distributed database system, however, a single database management system is discussed with respect to the example of FIG. 4. In an identification operation 410, a database management system identifies a plurality of transaction sequences based on a plurality of database queries. In this operation, each database query indicates one or more database transactions initiated by an application running on one of a plurality of clients in the distributed database system.

[0120] In some embodiments, each transaction sequence may be a continuous independent sequence or a linear time model that indicates database transactions from a personal point of view or the point of view of one or more applications running on a client. The personal point of view may be, for example, the point of view of a client system or an operator of the client system.

[0121] In some embodiments, the transaction sequences may be represented by a graph such as a causality graph or a serialization graph. Causality graphs and serialization graphs contain information about current and historic database transactions or operations, such as database queries received from a client system.

[0122] In some embodiments, serialization graph algorithms (SGAs) control the concurrent operation of temporally overlapping transactions by computing an equivalent serial ordering. SGAs try to "un-tangle" a convoluted sequence of operations by multiple transactions into a single cohesive thread of execution. SGAs function by creating a serialization
The nodes in the graph correspond to transactions in the system. The arcs of the graph correspond to equivalent serial ordering. As arcs are added to the graph, the algorithms look for cycles. If there are no cycles, then the transactions have an equivalent serial order and consistency is assured. If a serialization cycle were found, however, then consistency would be compromised if all transactions in the cycle were allowed to commit. In this case, the SGA would restore consistency by aborting one or more of the transactions forming the cycle.

In some embodiments, each causality graph represents the point of view of a client system, and thus the transaction sequence indicates all transaction initiated from that client. In other embodiments, each client system may have any number of associated transaction sequences. For example, a causality graph may represent the database transactions as perceived from an individual player of an online interactive game. Thus, the individual transaction sequences provide for the ability to eventually overlap database transactions by temporarily (during the read phase) taking into consideration only those database transactions relevant to that individual transaction sequence.

In a selection operation 420, the database management system selects a subset of the plurality of transaction sequences. In one embodiment, the selection of the subset of the plurality of transactions sequences may be chosen or otherwise selected by, for example, an application programmer in order to optimize performance as perceived by users. In some embodiments, the subset may be chosen arbitrarily or based on geography. In other embodiments, the subset may be chosen to indicate a group point of view or perception. For example, the subset of continuous independent sequences may be chosen based on social rank in a multi-user online interactive game. Ultimately, the choice of independent sequences in the subset impacts the direction and rate at which information propagates through the distributed database.

In a generation operation 430, the database management system generates an intermediate shared transaction sequence to continuously maintain transaction consistency among the subset of the plurality of transaction sequences. The intermediate shared transactions maintained in the intermediate shared transaction sequence are subsequently used to achieve global transaction consistency via a global transaction sequence that is replicated across a plurality of resources of the distributed database system.

Upon generation of the intermediate shared transaction sequence, in a maintenance operation 440, the database management system maintains the intermediate shared transaction sequence. In this example, maintaining the intermediate shared transaction sequence comprises asynchronously reconciling the subset of the plurality of transaction sequences into the intermediate shared transaction sequence. Advantageously, intermediate reconciliation of the uncommitted data transactions optimizes performance from the client-side perspective because barring any unresolvable inconsistency during reconciliation (i.e., during validation), from the client-side perspective the transaction appears to be committed to a global transaction sequence that is replicated across the distributed database system.

Lastly, in a commit operation 450, the database management system commits the previously uncommitted data transactions in the continuous shared sequence to a global transaction sequence.

FIGS. 5A and 5B depict diagrams illustrating examples of an intermediate reconciliation process in a distributed database system such as, for example, the distributed database 100 of FIG. 1. More specifically, FIGS. 5A and 5B illustrate how transactions are asynchronously and concurrently reconciled (and/or committed) to a global transaction sequence, according to an embodiment. The examples of FIGS. 5A and 5B are generally discussed with reference to an online gaming environment. However, it is appreciated that the online gaming environment may be any application running on client devices that generates database transactions.

Referring first to FIG. 5A which depicts an example process 510 that illustrates a typical optimistic concurrency control scheme for reconciliation without an intermediate shared transaction sequence. As discussed above, assertions can be used in lieu of locks to permit interleaving of operations and increase concurrency. In some embodiments, process 510 utilizes underlying assertions to interleave transactions received from the three transaction sequences.

More specifically, example process 510 illustrates the reconciliation process of three transaction sequences representing the database transactions initiated by applications running at each of three client systems, according to an embodiment. In this example, the transactions in each of the three transaction sequences can be triggered by operations of a user or player of an online interactive game or application; although the transactions may be triggered at the clients in other manners.

For example, as discussed above, in some embodiments optimistic schemes control concurrency by detecting invalid use after the fact by dividing a transaction’s existence into read, validate and publish phases. During the read phase, the transaction sequences for players 1, 2, and 3 acquire assumptions or assertions without regard to conflict or validity. The database management system and/or the transaction sequences themselves maintain a record of the set of assertions they use and set of assertions that they change. In some embodiments, assertions can be, for example, database key values.

As discussed above, a database transaction may include any number of assertions. During the validation phase, the assertions are examined to determine whether the current state of the assertion has changed. In typical distributed database systems, if the assertion has changed, then the assumptions or assertions upon which the database transaction relies are wrong and the system aborts the transaction. However, in the example of FIG. 5A, the assertions enforce consistency using various mechanisms such as, for example, a multi-version concurrency control (MVCC) mechanism. Thus, as disclosed herein, if the assumptions or assertions upon which the database transaction relies are wrong, then the concurrency control mechanisms in the database management system facilitate the ability to seek a time in the past during which assertions are true. This process is referred to herein as “time traveling.”

Once the assertions are validated, the system publishes the database transactions, committing the transaction’s changes during the commit (or publish) phase.

Referring now to FIG. 5B, which depicts an example illustrating an intermediate reconciliation process 520 as described herein, according to an embodiment.

In this example, database transactions initiated by the client systems associated with players of an online interactive game are asynchronously reconciled from multiple
specific points of view (i.e., players 1, and 2) to an intermediate shared point of view. The intermediate reconciliation process asynchronously and continuously reconciles among the transaction sequences associated with players 1 and 2. In some embodiments, the intermediate shared transaction sequence is then asynchronously and continuously reconciled with, for example, other players of the game (i.e., player 3) via their associated transaction sequences. The updates or changes to data items or database keys are then committed to the global transaction sequence for replication across the systems of the distributed database. Advantageously, the shared point of view or intermediate transaction sequence is reconciled relatively quickly (e.g., increased response time) from the perspective of players 1 and 2. The increased response time may provide players 1 and 2 with a better gaming experience.

[0136] FIGS. 6A and 6B depict transaction sequences illustrating example reconciliation processes 610 and 620 in a distributed database system, according to an embodiment. More specifically, the intermediate reconciliation processes 610 and 620 illustrate example reconciliations of the transaction sequences discussed with respect to FIG. 5A and FIG. 5B, respectively.

[0137] Referring first to FIG. 6A, which illustrates reconciliation of transaction sequences 610-1, 610-2, and 610-3 not using an intermediate shared transaction sequence, according to an embodiment. Transaction sequences 610-1, 610-2, and 610-3 include database transactions from associated players 1, 2, and 3 of FIG. 5A, respectively. As discussed above, the transactions sequences rely on underlying assertions that must be validated in order to reconcile and eventually commit the changes to a global transaction sequence 650.

[0138] The transaction sequences 610-1, 610-2, and 610-3 each illustrate the point of view of a client system or a user (or application) of a client system. For example, transaction sequences 610-1, 610-2, and 610-3 may illustrate linear time models of database transactions received from players 1, 2, and 3, respectively. In this example, database transactions C, D, E, and F are illustrated as included within transaction sequences 610-1, 610-2, and 610-3. The associated database transactions are labeled accordingly. For example, database transactions associated with transaction sequence 610-1 are labeled C1-F1, database transactions associated with transaction sequence 610-2 are labeled C2-F2, and database transactions associated with transaction sequence 610-3 are labeled C3-F3. Additionally, in this example, the oldest database transactions (e.g., the first to be received by the database management system) are shown on the left and the transactions are newest as you move to the right.

[0139] In this example, those database transactions that have been reconciled into the global transaction sequence 650A are shown with dotted lines. Unfortunately, the global reconciliation process can be rather time consuming from the perception of a user (or application) of the system resulting in the appearance of a slow system with slow response times. For example, a nominal delay of half a second to a second or more can give the appearance of a slow website or a slow online interactive gaming system.

[0140] Referring next to FIG. 6B, which illustrates reconciliation of transaction sequences 620-1 and 620-2 using an intermediate shared transaction sequence, followed by reconciliation of the intermediate shared transaction sequence with transaction sequence 620-3, according to an embodiment. Transaction sequences 620-1, 620-2, and 620-3 include database transactions from associated players 1, 2, and 3 of FIG. 5A, respectively. As discussed above, the transactions sequences rely on underlying assertions that must be validated in order to reconcile and eventually commit the changes to a global transaction sequence 650.

[0141] Introductions of the intermediate shared transaction sequence 640 improves (or at least provide the perception of) faster response times during reconciliation from the perspective of the first and second users because the shared transaction sequence need only be reconciled among transaction sequence 610-1 and 610-2, the reconciliation process can appear to take place quicker from the user’s perspective.

[0142] FIG. 7 depicts a diagram illustrating an example of an intermediate reconciliation process 700 in a distributed database system, according to an embodiment. More specifically, reconciliation process 700 illustrates reconciliation from various hierarchical levels (i.e., a multi-stage intermediate reconciliation process).

[0143] In this example, an application developer (e.g., online game developer) and/or the database management system may choose or select any number of intermediate shared transaction sequences allowing for control over the direction and flow of the reconciliation process. In operation, the database management system identifies the selected transaction sequences, generates the intermediate shared transaction sequences, and hierarchically maintains transaction consistency using the intermediate shared transaction sequences.

[0144] FIG. 8 depicts a flow diagram illustrating an example process 800 for hierarchically maintaining transaction consistency in a distributed database, according to an embodiment. One or more database management systems, such as, for example, the database management systems 18 of FIG. 1, among other functions, hierarchically maintain the transaction consistency in the distributed database system.

[0145] In operation 802, a database management system receives a query from a client system. As discussed above, the query can indicate one or more database transactions initiated by an application running on the client system. The distributed database system can receive any number of queries from any number of applications running on any number of client systems in the distributed database system; however, operation and handling of a single query is discussed in the example process 800 of FIG. 8.

[0146] In operation 804, the database management system processes the query to identify one or more assertions that require consensus among a plurality of machines (i.e., database resources or database management systems) within the distributed database in order to reconcile.

[0147] In operation 806, the database management system queries passive learners in the system to identify a history of the assertions as perceived from the passive learners. In some embodiments, the history of the assertions from the perspective of each of the passive learners represents, for example, the value they believe a database key to be for a time series (before and/or after specific database transactions). The history of changes to the assertion is kept by the passive learners so that the system can eventually determine the last time that there was a consensus among the machines (or database resources) on the value of an assertion or database key. This is discussed in greater detail with respect to operation 512.

[0148] In operation 808, the database management system determines whether or not a consensus exists among the resources with respect to the assertions relied upon by the one or more database transactions indicated in the query. If a
consensus exists then, in operation 810, the assertions can be drained into or toward the next transaction sequence at the next (or a higher) hierarchical level. In some cases, the next transaction sequence is a shared transaction sequence; however, the next transaction sequence may also be a global transaction sequence that is replicated across all of the machines in the distributed database system.

If a consensus does not exist then, in operation 812, the database management system falls back consistently across all assertions in the history of the passive learners until a consensus can be achieved. This process is referred to herein as “time traveling.” In operation 814, the system determines whether or not consensus is achieved among the resources with respect to the assertions relied upon by the one or more database transactions indicated in the query. If a consensus is achieved during the time traveling, then in operation 816 the database transactions that have a consensus are drained toward the next sequence and the other transaction sequences are removed.

If a consensus is not achieved through “time traveling” then, in process 818, the database management system determines whether a specific reconciliation procedure exits. If so, in process 820, the assertions are reconciled. However, if a consensus cannot be achieved using the history of the assertions as viewed from the passive learners and no specific reconciliation procedure exist then, in operation 822, the database management system aborts the transaction.

FIG. 9 shows a diagrammatic representation of a machine in the example form of a computer system 900 within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed.

In alternative embodiments, the machine operates as a standalone device or may be connected (e.g., networked) to other machines. In a networked deployment, the machine may operate in the capacity of a server or a client machine in a client-server network environment, or as a peer machine in a peer-to-peer (or distributed) network environment.

The machine may be a server computer, a client computer, a personal computer (PC), a tablet PC, a set-top box (STB), a personal digital assistant (PDA), a cellular telephone, a web appliance, a network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine.

While the machine-readable medium is shown in an exemplary embodiment to be a single medium, the term “machine-readable medium” shall be understood to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term “machine-readable medium” shall also be understood to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present invention.

In general, the routines executed to implement the embodiments of the disclosure, may be implemented as part of an operating system or a specific application, component, program, object, module or sequence of instructions referred to as “computer programs.” The computer programs typically comprise one or more instructions set at various times in various memory and storage devices in a computer, and that, when read and executed by one or more processors in a computer, cause the computer to perform operations to execute elements involving the various aspects of the disclosure.

Moreover, while embodiments have been described in the context of fully functioning computers and computer systems, those skilled in the art will appreciate that the various embodiments are capable of being distributed as a program product in a variety of forms, and that the disclosure applies equally regardless of the particular type of machine or computer-readable media used to actually effect the distribution.

Further examples of machine or computer-readable media include but are not limited to recordable type media such as volatile and non-volatile memory devices, floppy and other removable disks, hard disk drives, optical disks (e.g., Compact Disk Read-Only Memory (CD ROMs), Digital Versatile Disks (DVDs), etc.), among others, and transmission type media such as digital and analog communication links.

Unless the context clearly requires otherwise, throughout the description and the claims, the words “comprise,” “comprising,” and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense; that is to say, in the sense of “including, but not limited to.” As used herein, the terms “connected,” “coupled,” or any variant thereof, means any connection or coupling, either direct or indirect, between two or more elements; the coupling of connection between the elements can be physical, logical, or a combination thereof. Additionally, the words “herein,” “above,” “below,” and words of similar import, when used in this application, shall refer to this application as a whole and not to any particular portions of this application. Where the context permits, words in the above Detailed Description using the singular or plural number may also include the plural or singular number respectively. The word “or,” in reference to a list of two or more items, covers all of the following interpretations of the word: any of the items in the list, all of the items in the list, and any combination of the items in the list.

The above detailed description of embodiments of the disclosure is not intended to be exhaustive or to limit the teachings to the precise form disclosed above. While specific embodiments of, and examples for, the disclosure are described above for illustrative purposes, various equivalent modifications are possible within the scope of the disclosure, as those skilled in the relevant art will recognize. For example, while processes or blocks are presented in a given order, alternative embodiments may perform routines having steps, or employ systems having blocks, in a different order, and some processes or blocks may be deleted, moved, added, subdivided, combined, and/or modified to provide alternative or subcombinations. Each of these processes or blocks may be implemented in a variety of different ways. Also, while processes or blocks are at times shown as being performed in series, these processes or blocks may instead be performed in parallel, or may be performed at different times. Further any specific numbers noted herein are only examples: alternative implementations may employ differing values or ranges.

The teachings of the disclosure provided herein can be applied to other systems, not necessarily the system described above. The elements and acts of the various embodiments described above can be combined to provide further embodiments.

Any patents and applications and other references noted above, including any that may be listed in accompany-
ing filing papers, are incorporated herein by reference. Aspects of the disclosure can be modified, if necessary, to employ the systems, functions, and concepts of the various references described above to provide yet further embodiments of the disclosure.

[0162] These and other changes can be made to the disclosure in light of the above Detailed Description. While the above description describes certain embodiments of the disclosure, and describes the best mode contemplated, no matter how detailed the above appears in text, the teachings can be practiced in many ways. Details of the system may vary considerably in its implementation details, while still being encompassed by the subject matter disclosed herein. As noted above, particular terminology used when describing certain features or aspects of the disclosure should not be taken to imply that the terminology is being redefined herein to be restricted to any specific characteristics, features, or aspects of the disclosure with which that terminology is associated. In general, the terms used in the following claims should not be construed to limit the disclosure to the specific embodiments disclosed in the specification, unless the above Detailed Description section explicitly defines such terms. Accordingly, the actual scope of the disclosure encompasses not only the disclosed embodiments, but also all equivalent ways of practicing or implementing the disclosure under the claims.

[0163] While certain aspects of the disclosure are presented below in certain claim forms, the inventors contemplate the various aspects of the disclosure in any number of claim forms. For example, while only one aspect of the disclosure is recited as a means-plus-function claim under 35 U.S.C. §112, ¶6, other aspects may likewise be embodied as a means-plus-function claim, or in other forms, such as being embodied in a computer-readable medium. (Any claims intended to be treated under 35 U.S.C. §112, ¶6 will begin with the words “means for.”) Accordingly, the applicant reserves the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the disclosure.

What is claimed is:

1. A method of hierarchically maintaining transaction consistency in a distributed database, the method comprising: identifying, at a database management system, a plurality of transaction sequences based on a plurality of database queries, wherein each database query indicates one or more database transactions initiated by an application running on one of a plurality of clients in the distributed database; selecting, at the database management system, a subset of the plurality of transaction sequences; and generating, at the database management system, an intermediate shared transaction sequence to continuously maintain transaction consistency among the subset of the plurality of transaction sequences, wherein intermediate shared transactions maintained in the intermediate shared transaction sequence are subsequently used to achieve global transaction consistency via a global transaction sequence that is replicated across a plurality of resources of the distributed database.

2. The method of claim 1, further comprising: replicating the global transaction sequence across the plurality of resources of the distributed database.

3. The method of claim 1, wherein each transaction sequence of the subset of the plurality of transaction sequences indicates a causal history of database transactions from the perspective of one of the applications.

4. The method of claim 1, further comprising: maintaining the intermediate shared transaction sequence, wherein maintaining the intermediate shared transaction sequence comprises asynchronously reconciling the subset of the plurality of transaction sequences into the intermediate shared transaction sequence.

5. The method of claim 4, wherein each database transaction includes one or more assertions and reconciling the subset of the plurality of transaction sequences into the intermediate shared transaction sequence comprises determining the validity of each assertion.

6. The method of claim 5, wherein determining the validity of each assertion comprises moving consistently within each transaction sequence from a source transaction to a cause transaction until each assertion is validated.

7. The method of claim 1, wherein the intermediate shared transaction sequence represents a shared point of view as perceived from two or more applications operating on two or more clients of the plurality of clients.

8. The method of claim 1 wherein one or more transaction sequences of the subset of the plurality of transaction sequences are selected based on the applications that initiated the one or more transaction sequences.

9. The method of claim 1, wherein one or more transaction sequences of the subset of the plurality of transaction sequences are selected based on geographic locations of one or more clients associated with the one or more transaction sequences.

10. The method of claim 1 wherein one or more transaction sequences of the subset of the plurality of transaction sequences are selected based on an attribute of one or more of the plurality of clients associated with the one or more transaction sequences.

11. The method of claim 1, further comprising: committing the shared transactions in the intermediate shared transaction sequence to maintain the global transaction sequence, wherein committing the shared transactions to the global transaction sequence comprises reconciling one or more of the shared transactions in the intermediate shared transaction sequence with other database transactions in the distributed database.

12. The method of claim 10, wherein each database transaction includes one or more assertions and reconciling comprises achieving consensus among a plurality of database resources regarding the validity of each assertion.

13. The method of claim 12, wherein achieving the consensus comprises moving consistently within each transaction sequence from a source database transaction to cause database transaction until each assertion is validated.

14. The method of claim 10, further comprising: prior to committing the shared transactions to the global transaction sequence, notifying one of the applications that the associated database query is completed.

15. The method of claim 10, further comprising: committing other uncommitted database transactions of the plurality of database transactions to the global transaction sequence, wherein the other uncommitted database transactions are not in the intermediate shared transaction sequence.

16. The method of claim 1, wherein the subset of the plurality of transaction sequences is selected based on a relation between the users of a first application.
17. The method of claim 16, wherein each user has a user profile associated with the first application and wherein the subset of the plurality of transactions sequences is selected based on a relation between the user profiles.

18. The method of claim 16, wherein the subset of the plurality of transaction sequences is selected based on a type of the first application.

19. The method of claim 16, wherein the first application comprises a multi-user online interactive game.

20. The method of claim 19, wherein the subset of the plurality of transaction sequences is selected based on a social rank in the multi-user online interactive game.

21. A database management system comprising:

a processing unit;

an interface configured to receive a plurality of database queries, wherein each database query indicates one or more database transactions initiated by an application running on one of a plurality of clients in a distributed database system;

a memory unit having instructions stored thereon, wherein the instructions, when executed by the processing unit, cause the processing unit to identify a plurality of transaction sequences based on the plurality of database queries, select a subset of the plurality of transaction sequences, and generate an intermediate shared transaction sequence to maintain transaction consistency among the subset of the plurality of transaction sequences.

22. The database management system of claim 21, wherein intermediate shared transactions maintained in the intermediate shared transaction sequence are subsequently used to achieve global transaction consistency via a global transaction sequence that is replicated across a plurality of resources of the distributed database.

23. The database management system of claim 21, wherein the plurality of resources comprise other database management systems in the distributed database system.

24. The database management system of claim 21, wherein the plurality of resources comprises storage management systems in the distributed database system.

25. The database management system of claim 21, wherein each transaction sequence of the subset of the plurality of transaction sequences indicates a causal history of database transactions from the perspective of one of the applications.

26. The database management system of claim 21, wherein the intermediate shared transaction sequence represents a shared point of view as perceived from two or more applications operating on two or more clients of the plurality of clients.

27. The database management system of claim 21, wherein one or more transaction sequences of the subset of the plurality of transaction sequences are selected based on the applications that initiated the one or more transaction sequences.

28. The database management system of claim 21, wherein one or more transaction sequences of the subset of the plurality of transaction sequences are selected based on geographic locations of one or more clients associated with the one or more transaction sequences.

29. The database management system of claim 21, wherein one or more transaction sequences of the subset of the plurality of transaction sequences are selected based on an attribute of one or more of the plurality of clients associated with the one or more transaction sequences.

30. A method of hierarchically maintaining transaction consistency in a distributed database system, the method comprising:

receiving, at the database management system, a plurality of database transactions from a plurality of client systems in the distributed database system, wherein each transaction sequence indicates a transaction initiated by an application running on one of a plurality of clients;

identifying, at the database management system, a plurality of transaction sequences based on the plurality of database transactions, wherein each database transaction is initiated by an application running on one of a plurality of clients in the distributed database system;

selecting, at the database management system, a subset of the plurality of transaction sequences based on a first criteria;

generating, at the database management system, an intermediate shared transaction sequence to maintain transaction consistency among the subset of the plurality of transaction sequences;

 Committing the database transactions indicated by the intermediate shared transaction sequence to a global transaction sequence; and

prior to committing the database transactions indicated by the intermediate shared transaction sequence to the global transaction sequence, sending, from the database management system, a notification indicating a commit or failure of one or more of the database transactions in the intermediate shared transaction sequence to the application on client device that initiated the database request.

31. The method of claim 30, wherein the first criteria comprises an association between the applications that initiated the subset of the plurality of transaction sequences.

32. A database management system comprising:

means for identifying a plurality of transaction sequences based on a plurality of database queries, wherein each database query indicates one or more database transactions initiated by an application running on one of a plurality of clients in the distributed database;

means for selecting a subset of the plurality of transaction sequences; and

means for generating an intermediate shared transaction sequence to maintain transaction consistency among the subset of the plurality of transaction sequences.

*    *    *    *    *