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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 June 2016 (02.06.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/085488 A1

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

(21) International Application Number:
PCT/US2014/067599

(22) International Filing Date:
26 November 2014 (26.11.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **SIMITSIS, Alkiviadis**; 1501 Page Mill Rd, Palo Alto, California 94304-1100 (US). **WILKINSON, William K**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **ARORA, Vaibhav**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US).

(74) Agents: **ESTEBAN, Michelle** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (US).

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

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

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

Published:

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

(54) Title: DATABASE TRANSFER OF CHANGES

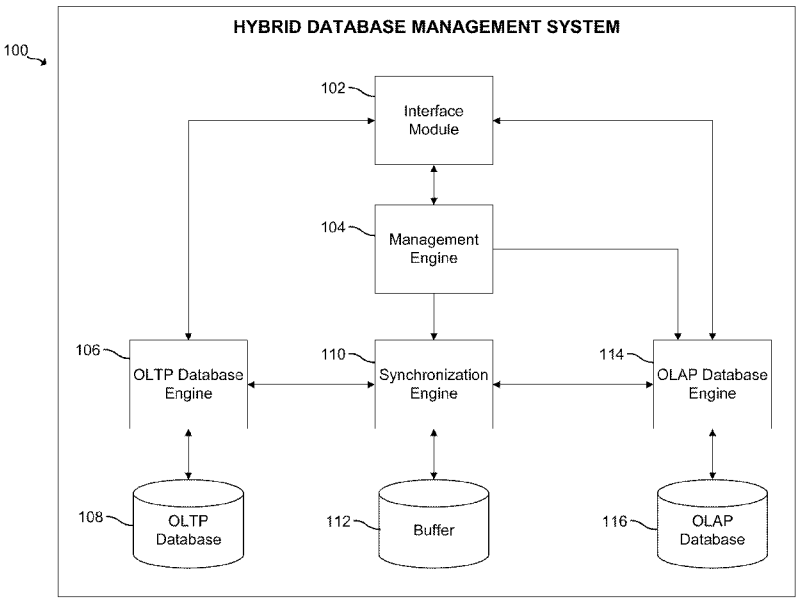

FIG. 1

(57) Abstract: Example implementations relate to a database transfer of changes. For example, a computing device may include at least one processor. The at least one processor may receive a stream of changes to an online transaction processing (OLTP) database and may cache the stream of changes in a buffer. The at least one processor may identify specified criteria indicating a manner of sending the stream of changes to an online analytical processing (OLAP) database and may transfer the stream of changes from the buffer to the OLAP database based on the specified criteria.

WO 2016/085488 A1

DATABASE TRANSFER OF CHANGES

BACKGROUND

[0001] Many entities (e.g., enterprises, organizations, computer applications, etc.) utilize databases for storage of data relating to the entities. The data in a database may be received from a data stream of incoming data. Data stored in these databases may be accessed and analyzed for various purposes.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Some examples of the present application are described with respect to the following figures:

[0003] FIG. 1 is a block diagram of an example hybrid database management system for transferring changes to an online analytical processing database;

[0004] FIG. 2 is a block diagram of a computing device for transferring changes to an online analytical processing database; and

[0005] FIG. 3 is a flowchart illustrating an example method of transferring changes to an online analytical processing database.

DETAILED DESCRIPTION

[0006] As described above, data stored in a database may be accessed and analyzed for various purposes. A database management system (DBMS) may manage and control access to a particular database in response to queries for data. Typically, a DBMS may be optimized for a particular type of workload, such as online transaction processing (OLTP) or online analytical processing (OLAP). OLTP pertains to a class of information systems that may facilitate and manage transaction-oriented applications, such as data entry and retrieval transaction processing. OLAP is an approach to answering multi-dimensional analytical queries, such as business reporting. OLTP requests may be relatively short and may read or write only a few database

records, while OLAP requests may be relatively long, may access a large number of records, and may allow primarily read-only access. As such, OLAP requests sent to an OLTP-optimized DBMS and/or OLTP requests sent to an OLAP-optimized DBMS may generally perform poorly.

[0007] A hybrid DBMS having a synchronization engine may be utilized to optimize access to and/or modification of both an OLTP database and an OLAP database, providing a unified framework capable of handling both OLTP and OLAP workloads concurrently. In some examples, the features of the hybrid DBMS may be implemented as a module on top of existing OLTP and OLAP DBMSs. The OLTP and OLAP databases contain at least some common data, but the common data in each database may be stored in different representations, where the common data in the OLTP database may be the current version of the common data while the common data in the OLAP database may be either the current version or a previous version of the common data. The synchronization engine of the hybrid DBMS may manage the synchronization of modifications from the OLTP database to the OLAP database to ensure that modifications to the OLTP database are propagated to the OLAP database and may provide access to data that is in-transit between the OLTP database and the OLAP database. An interface module of the hybrid DBMS may be used to interface the hybrid DBMS with one or more applications such that the hybrid DBMS may appear to the applications as a single DBMS. In some examples, the hybrid DBMS is not a federated system, and the interface module may control access and/or updates to the OLTP and the OLAP database engines of the hybrid DBMS. As such, any applications accessing and/or updating the OLTP and OLAP databases through the interface module do not directly access the OLTP and the OLAP database engines.

[0008] The synchronization engine of the hybrid DBMS may receive a stream of committed changes to the OLTP database and may cache the stream of changes in a local buffer. The synchronization engine may load the buffered changes into the OLAP database according to specified criteria indicating a manner of sending the buffered changes to the OLAP database. The specified criteria may be default criteria and/or user-specified criteria that

may indicate application-specified objectives (e.g., freshness of data, throughput, periodic transfer of data, etc.) relating to a manner of sending buffered changes to the OLAP database. Loading of buffered changes may be transactionally consistent such that when a set of transactions is committed by the OLTP database engine, the entire set of committed transactions is sent to the OLAP database engine for storage in the OLAP database. The synchronization engine may provide query access over the buffer such that applications may be able to query the freshest data.

[0009] The hybrid DBMS may direct OLTP transactions to the OLTP DBMS and may direct OLAP transactions to the OLAP DBMS, which may provide efficient performance because each request received may be sent to the DBMS that is optimized to process it. Modifications (e.g., updates, deletions, insertions), even those within an OLAP transaction, may be directed to the OLTP DBMS. These modifications are propagated from the OLTP DBMS to the OLAP DBMS in a manner that preserves database consistency and keeps pace with the OLTP DBMS. Additionally, transaction isolation is ensured despite transactions that span multiple DBMSs. Techniques to ensure consistency and isolation are described in further detail below. While the examples disclosed herein describe a hybrid architecture of a single OLTP database engine and a single OLAP database engine, one of ordinary skill in the art will appreciate that the techniques disclosed herein may be used with multiple independent OLTP database engines and a single OLAP database engine. For example, to increase transaction throughput, an OLTP database may be partitioned across multiple independent OLTP database engines where each OLTP transaction may access a single partition. The hybrid DBMS described herein may support this partitioning for multiple independent OLTP database engines. For example, an update transaction at a particular OLTP database engine may access a single OLTP database partition as well as the OLAP database, while a read-only transaction may access any partition as well as the OLAP database.

[0010] Referring now to the figures, FIG. 1 is a block diagram of an example hybrid DBMS 100 for transferring changes to an OLAP database.

The components of hybrid DBMS 100 may operate using one or more processors (not shown) to perform the functions of the components.

[0011] Interface module 102 is a hardware-implemented and/or processor-implemented module that provides one or more unified application programming interfaces (APIs) to any applications to allow communication between the applications and hybrid DBMS 100. Interface module 102 may maintain session and transaction context, accept requests for data, forward those requests to the appropriate DBMS (e.g., OLTP database engine 106 or OLAP database engine 114) for processing, and return results in response to the requests.

[0012] OLTP database engine 106 is a hardware-implemented and/or processor-implemented module that manages and controls writing data to OLTP database 108, reading data from OLTP database 108, and processing OLTP requests. OLTP database 108 may be any suitable database optimized for OLTP. While examples disclosed herein describe an OLTP database engine and an OLTP database, one of ordinary skill in the art will recognize that any suitable write-optimized database engine and write-optimized database may be used with the techniques described herein.

[0013] OLAP database engine 114 is a hardware-implemented and/or processor-implemented module that manages and controls writing data to OLAP database 116, reading data from OLAP database 116, and processing OLAP requests. OLAP database 116 may be any suitable database optimized for OLAP. While examples disclosed herein describe an OLAP database engine and an OLAP database, one of ordinary skill in the art will recognize that any suitable read-optimized database engine and read-optimized database may be used with the techniques described herein.

[0014] Synchronization engine 110 is a hardware-implemented and/or processor-implemented module that may manage and control the synchronization of data between OLTP database engine 106 and OLAP database engine 114. For example, synchronization engine 110 may collect changes to table rows in OLTP database 108 from OLTP database engine 106, cache the changes locally in buffer 112, and load the changes to the

OLAP database engine 114 for storage in OLAP database 116 at the appropriate time and/or in the appropriate manner based on specified criteria. Buffer 112 may be any suitable storage device capable of storing changes from OLTP database engine 106. Synchronization engine 110 may provide query capability over in-transit data that may be stored in buffer 112 but not yet loaded to OLAP database engine 114. Synchronization engine 110 may also validate transactions.

[0015] Management engine 104 is a hardware-implemented and/or processor-implemented module that provides various management functions, such as managing criteria specifying a manner of sending changes from buffer 112 to OLAP database engine 114 for storage in OLAP database 116, determining when and/or how to initiate transfer of changes from buffer 112 to OLAP database engine 114 for storage in OLAP database 116, managing garbage collection of old data, and the like. For example, management engine 104 may manage criteria that specifies a particular time delay for collecting garbage (e.g., collect every 5 seconds, collect when fresh data is requested by a query, etc.). Garbage collection refers to a processing of collecting data that may no longer be requested by any query (e.g., old versions of data that has since been updated).

[0016] As described earlier, while read requests may be sent to either OLTP database engine 106 or OLAP database engine 114, data modification requests (e.g., insert, delete, and/or update operations) are directed to OLTP database engine 106. In either case, hybrid DBMS 100 provides each request with a consistent view of the databases associated with hybrid DBMS 100 to ensure that each transaction sees a consistent view of the data. If the transaction is a single request, consistency is ensured because the request may be entirely processed by one engine. For example, assuming each transaction executes entirely on OLTP database engine 106 or entirely on OLAP database engine 114, OLTP database engine 106 may perform local concurrency control using any scheme it chooses, resulting in transactions that are isolated and serializable such that transactions appear to execute sequentially and one at a time. In this case, when a transaction is ready to commit, OLTP database engine 106 may send a transaction validation

request to synchronization engine 110 and may include the transaction start time and an identification of the rows modified during the transaction. As such, synchronization engine 110 may be able to immediately acknowledge the commit to OLTP database engine 106, and no validation may be required since there may be no conflicting transactions.

[0017] Read-only queries that execute only on OLAP database engine 114 may also be provided a consistent view of OLTP database 108 and OLAP database 116 because, as described earlier, data changes from OLTP database engine 106 are applied to OLAP database engine 114 as logically atomic, transactionally consistent operations. If a transaction includes multiple requests, consistency may be ensured. For example, assuming that multi-request transactions are delimited by begin and commit requests, the begin request may identify the database engine that may handle the transaction (e.g., OLTP database engine 106 for transactions that modify data, either OLTP database engine 106 or OLAP database engine 114 for read-only transactions, etc.). When an OLAP multi-statement transaction statement begins, if the transaction requests the freshest data, the transaction waits until synchronization engine 110 sends the latest batch of buffered updates. Otherwise, the transaction may begin without waiting. OLAP database engine 114 then guarantees the transaction a consistent view of the data for all its statements until it commits. For example, the view may be read-committed, serializable, and the like, depending on application requirements. For a multi-statement OLTP transaction (e.g., a read-only query sent to OLAP database engine 114 to compute some aggregate value over a set of data objects), OLTP database engine 106 may not see the read-only query sent to OLAP database engine 114, so the OLTP database engine 106 may not, by itself, ensure the transaction sees a consistent view of data it accessed. To address this, synchronization engine 110 may perform a validation operation as part of the transaction commit operation to ensure that the data read by the OLAP query has not been updated. Validation may be performed is several ways, as described below.

[0018] The row-level data modifications of a transaction may be cached in buffer 112 by synchronization engine 110. Additionally, views may be

installed on OLAP database engine 114 that enable this cached data to be queried on-demand to obtain the most recent data. For example, if an application submits an OLAP query that requests fresh data (e.g., the most recent data), OLAP database engine 114 may utilize synchronization engine 110 to read the cached data in buffer 112. While this may not provide an application with a consistent view of data, it may be made available as an application option. Periodically (or at any other time indicated by the specified criteria), management engine 104 may signal synchronization engine 110 to forward a consistent set of cached updates to OLAP database engine 114. At this time, synchronization engine 110 may merge the cached modifications into a single batch update transaction that may be sent to OLAP database engine 114 to synchronize the data in OLAP database 116 with data in OLTP database 108. OLAP database engine 114 may process and commit this batch update before the arrival of the next batch update from synchronization engine 110 without interference with concurrent analytic queries on OLAP database engine 114. Once OLAP database engine 114 has acknowledged the batch update, the cached data that was transferred may be purged from buffer 112. As such, when a query of OLAP database engine 114 requests fresh data, OLAP database engine 114 may request changes from synchronization engine 110, and synchronization engine 110 may send the requested changes to OLAP database engine 114. The query may then be processed using OLAP database engine 114.

[0019] In some examples, a split transaction may be processed on hybrid DBMS 100. A split transaction may be a multi-statement transaction in which some statements execute on OLTP database engine 106 and some statements execute on OLAP database engine 114. For example, an OLTP transaction may request aggregate values computed over a large amount of historical data in order to determine how to modify the database. Such a query may be more efficiently processed by OLAP database engine 114 rather than OLTP database engine 106, and a split transaction may be generated. Because OLTP database engine 106 is unaware of any statements processed by OLAP database engine 114, read-write conflicts between changes on OLTP database engine 106 and queries on OLAP

database engine 114 may be missed. To detect these conflicts, synchronization engine 110 may perform a validation operation. Table 1 below summarizes examples of possible conflicts, where $Xact_{val}$ refers to a transaction to be validated and $Xact_{cmt}$ refers to another transaction that has committed during the lifetime of $Xact_{val}$. A read operation of row X is denoted as Rd X, and a write operation of row X is denoted Wt X. To maintain database consistency, if a first transaction commits before a second transaction, then any database changes made by the first transaction are visible to the second transaction.

Case No.	$Xact_{cmt}$	$Xact_{val}$
1	Rd X	Rd X
2	Rd X	Wt X
3	Wt X	Rd X
4	Wt X	Wt X

Table 1. Examples of Validating Transactions

[0020] In example case number 1 in Table 1 above, the read operation of $Xact_{cmt}$ does not conflict with the read operation of $Xact_{val}$ because committing a read of row X during $Xact_{val}$ does not conflict with performing a read of row X. As such, $Xact_{val}$ is successfully validated.

[0021] Similarly, in example case number 2 in Table 1 above, the read operation of $Xact_{cmt}$ does not conflict with the write operation of $Xact_{val}$ because committing a read of row X during $Xact_{val}$ does not conflict with performing a write of row X. As such, $Xact_{val}$ is successfully validated.

[0022] In example case number 4 in Table 1 above, the write operation of $Xact_{cmt}$ and $Xact_{val}$ may be detected and handled by OLTP database engine 106, and as such, $Xact_{val}$ is successfully validated.

[0023] In example case number 3 in Table 1 above, a possible conflict may arise depending on which version of X was read by $Xact_{val}$. If $Xact_{val}$

read the row X version written by $Xact_{cmt}$, there is no conflict, and $Xact_{val}$ may commit. If $Xact_{val}$ read the row X version prior to the $Xact_{cmt}$ write operation, then the transactions are not serializable, and $Xact_{val}$ cannot commit. To detect this conflict, a validation operation may be performed in any suitable manner. For example, it may be possible to compute and retain a content-based hash of the values read. Then, at validation, the read operation may be repeated, and a content-based hash of that result may be determined. This hash may be compared with the previously hashed value. If the values are the same, $Xact_{val}$ is successfully validated. If the values are different, validation fails.

[0024] Management engine 104 may manage the task of computing and retaining the content-based hash result of split-transaction statements sent to OLAP database engine 114. At validation time, these statements and their hashes are sent from management engine 104 to synchronization engine 110 for validation by synchronization engine 110. For example, in example case number 3 in Table 1 above, a content-based hash may be computed on the value of X read by $Xact_{val}$ and retained by management engine 104. At validation time, management engine 104 may send this content-based hash to synchronization engine 110 along with the corresponding query that read row X. To validate that the read result has not changed, synchronization engine 110 executes this query again and computes the hash on the current result set. If the has value matches that of the initial read query, validation is successful and $Xact_{val}$ may commit. The read operation for validation may read data committed while $Xact_{val}$ was executing. To do this, OLAP database engine 114 may read from synchronization engine 110 any committed updates that have not yet been sent to OLAP database engine 114. While a particular validation operation is described herein, one of ordinary skill in the art will appreciate that other validation techniques may be used. For example, the validation operation may include comparing read and write sets.

[0025] FIG. 2 is a block diagram of an example computing device 200 for transferring changes to an OLAP database. In some examples, computing device 200 may be a synchronization engine, such as synchronization engine 110 of FIG. 1.

[0026] Computing device 200 may be, for example, a web-based server, a local area network server, a cloud-based server, a notebook computer, a desktop computer, an all-in-one system, a tablet computing device, a mobile phone, an electronic book reader, a printing device, or any other electronic device suitable for transferring changes to an OLAP database. Computing device 200 may include a processor 202 and a machine-readable storage medium 204. Computing device 200 may store a stream of changes to an OLTP database in a buffer and may transmit the stream of changes from the buffer to an OLAP database based on criteria specifying a manner of sending the stream of changes to the OLAP database.

[0027] Processor 202 is a tangible hardware component that may be a central processing unit (CPU), a semiconductor-based microprocessor, and/or other hardware devices suitable for retrieval and execution of instructions stored in machine-readable storage medium 204. Processor 202 may fetch, decode, and execute instructions 206, 208, 210, and 212 to control a process of transferring changes to an OLAP database. As an alternative or in addition to retrieving and executing instructions, processor 202 may include at least one electronic circuit that includes electronic components for performing the functionality of instructions 206, 208, 210, 212, or a combination thereof.

[0028] Machine-readable storage medium 204 may be any electronic, magnetic, optical, or other physical storage device that contains or stores executable instructions. Thus, machine-readable storage medium 204 may be, for example, Random Access Memory (RAM), an Electrically Erasable Programmable Read-Only Memory (EEPROM), a storage device, an optical disc, and the like. In some examples, machine-readable storage medium 204 may be a non-transitory storage medium, where the term “non-transitory” does not encompass transitory propagating signals. As described in detail below, machine-readable storage medium 204 may be encoded with a series of processor executable instructions 206, 208, 210, and 212 for receiving a stream of changes to an OLTP database, storing the stream of changes in a buffer, identifying specified criteria indicating a manner of sending the stream of changes to an OLAP database, and transmitting the stream of changes from the buffer to the OLAP database based on the specified criteria.

[0029] Stream receipt instructions 206 may manage and control receipt of a stream of changes. The stream of changes may be changes to be applied to an OLTP database. For examples, the stream of changes may include updates, insertions, and/or deletions associated with the OLTP database.

[0030] Buffer storage instructions 208 may manage and control the storing of the stream of changes in a buffer. For example, buffer storage instructions 208 may store the stream of changes in the buffer after the stream of changes has been received.

[0031] Criteria identification instructions 210 may manage and control the identification of specified criteria indicating a manner of sending the stream of changes to an OLAP database. The specified criteria may be any suitable criteria indicating the manner of sending the stream of changes to the OLAP database, such as how to send the stream of changes, when to send the stream of changes (e.g., periodically, based on availability, etc.), an amount and/or size to send to the OLAP database, and the like.

[0032] Stream transmission instructions 212 may manage and control the transmission of the stream of changes from the buffer to the OLAP database. For example, stream transmission instructions 212 may manage and control the transmission of the stream of changes based on the specified criteria identified.

[0033] FIG. 3 is a flowchart illustrating an example method 300 of transferring changes to an OLAP database. Method 300 may be implemented using computing device 200 of FIG. 2.

[0034] Method 300 includes, at 302, receiving a stream of changes to an OLTP database. The stream of changes may include any changes to be applied to the OLTP database, such as updates, insertions, deletions, and the like.

[0035] Method 300 also includes, at 304, storing the stream of changes in a buffer. The stream of changes may be stored in the buffer after the stream of changes has been received.

[0036] Method 300 also includes, at 306, determining specified criteria indicating a manner of sending the stream of changes to an OLAP database.

The specified criteria may be any suitable criteria indicating the manner of sending the stream of changes to the OLAP database, such as how and/or when to send the stream of changes, an amount and/or size to send to the OLAP database, and the like.

[0037] Method 300 also includes, at 308, sending the stream of changes from the buffer to the OLAP database based on the specified criteria. For example, if the specified criteria indicates that any changes in the buffer are to be sent to the OLAP after a particular amount of time has elapsed since the last data transfer, the stream of changes may be sent from the buffer to the OLAP database after that particular amount of time has elapsed.

[0038] Examples provided herein (e.g., methods) may be implemented in hardware, software, or a combination of both. Example systems may include a controller/processor and memory resources for executing instructions stored in a tangible non-transitory medium (e.g., volatile memory, non-volatile memory, and/or machine-readable media). Non-transitory machine-readable media can be tangible and have machine-readable instructions stored thereon that are executable by a processor to implement examples according to the present disclosure.

[0039] An example system can include and/or receive a tangible non-transitory machine-readable medium storing a set of machine-readable instructions (e.g., software). As used herein, the controller/processor can include one or a plurality of processors such as in a parallel processing system. The memory can include memory addressable by the processor for execution of machine-readable instructions. The machine-readable medium can include volatile and/or non-volatile memory such as a random access memory ("RAM"), magnetic memory such as a hard disk, floppy disk, and/or tape memory, a solid state drive ("SSD"), flash memory, phase change memory, and the like.

Claims

What is claimed is:

1. A system comprising:

at least one processor to:

receive a stream of changes to an online transaction processing (OLTP) database;

cache the stream of changes in a buffer;

identify specified criteria indicating a manner of sending the stream of changes to an online analytical processing (OLAP) database; and

transfer the stream of changes from the buffer to the OLAP database based on the specified criteria.

2. The computing device of claim 1, wherein the specified criteria indicates periodically sending the stream of changes to the OLAP database.

3. The computing device of claim 1, wherein the stream of changes includes at least one of an update, an insertion, and a deletion.

4. The computing device of claim 1, wherein the at least one processor is further to:

receive a query of the OLAP database, the query requesting fresh data;

send the fresh data from the buffer to the OLAP database in response to the query; and

process the query using the OLAP database after the fresh data is sent to the OLAP database.

5. The computing device of claim 1, wherein the at least one processor is further to:

receive a query of the OLTP database, the query requesting historic data;

validate changes sent from the OLTP database to the OLAP database;

generate an OLTP sub-query and an OLAP sub-query relating to the query, the OLTP sub-query and the OLAP sub-query being generated based on the changes being validated; and

process the OLTP sub-query and an OLAP sub-query.

6. A method comprising:

receiving, by a computing device, a stream of changes to an online transaction processing (OLTP) database, the stream of changes including at least one of an update, an insertion, and a deletion with respect to an online analytical processing (OLAP) database;

storing, by the computing device, the stream of changes in a buffer;

determining, by the computing device, specified criteria indicating a manner of sending the stream of changes to the OLAP database; and

sending, by the computing device, the stream of changes from the buffer to the OLAP database based on the specified criteria.

7. The method of claim 6, wherein the specified criteria indicates periodically sending the stream of changes to the OLAP database.

8. The method of claim 6, wherein the specified criteria indicates sending the stream of changes to the OLAP database based on availability of the OLAP database.

9. The method of claim 6, further comprising:

receiving, by the computing device, a query of the OLAP database, the query requesting fresh data;

sending, by the computing device, the fresh data from the buffer to the OLAP database in response to the query; and

processing, by the computing device, the query using the OLAP database after the fresh data is sent to the OLAP database.

10. The method of claim 6, further comprising:

receiving, by the computing device, a query of the OLTP database, the query requesting historic data;

validating, by the computing device, changes sent from the OLTP database to the OLAP database;

generating, by the computing device, an OLTP sub-query and an OLAP sub-query relating to the query, the OLTP sub-query and the OLAP sub-query being generated based on the changes being validated; and

processing, by the computing device, the OLTP sub-query and an OLAP sub-query.

11. A non-transitory machine-readable storage medium storing instructions that, if executed by at least one processor of a computing device, cause the computing device to:

receive a stream of changes to an online transaction processing (OLTP) database;

store the stream of changes in a buffer;

identify specified criteria indicating a manner of sending the stream of changes to an online analytical processing (OLAP) database, the OLAP database having data previously stored in the OLTP database; and

transmit the stream of changes from the buffer to the OLAP database based on the specified criteria.

12. The non-transitory machine-readable storage medium of claim 11, wherein the specified criteria indicates periodically sending the stream of changes to the OLAP database.

13. The non-transitory machine-readable storage medium of claim 11, wherein the stream of changes includes at least one of an update, an insertion, and a deletion.

14. The non-transitory machine-readable storage medium of claim 11, wherein the instructions further cause the computing device to:

receive a query of the OLAP database, the query requesting fresh data;

send the fresh data from the buffer to the OLAP database in response to the query; and

process the query using the OLAP database after the fresh data is sent to the OLAP database.

15. The non-transitory machine-readable storage medium of claim 11, wherein the instructions further cause the computing device to:

receive a query of the OLTP database, the query requesting historic data;

validate changes sent from the OLTP database to the OLAP database;

generate an OLTP sub-query and an OLAP sub-query relating to the query, the OLTP sub-query and the OLAP sub-query being generated based on the changes being validated; and

process the OLTP sub-query and an OLAP sub-query.

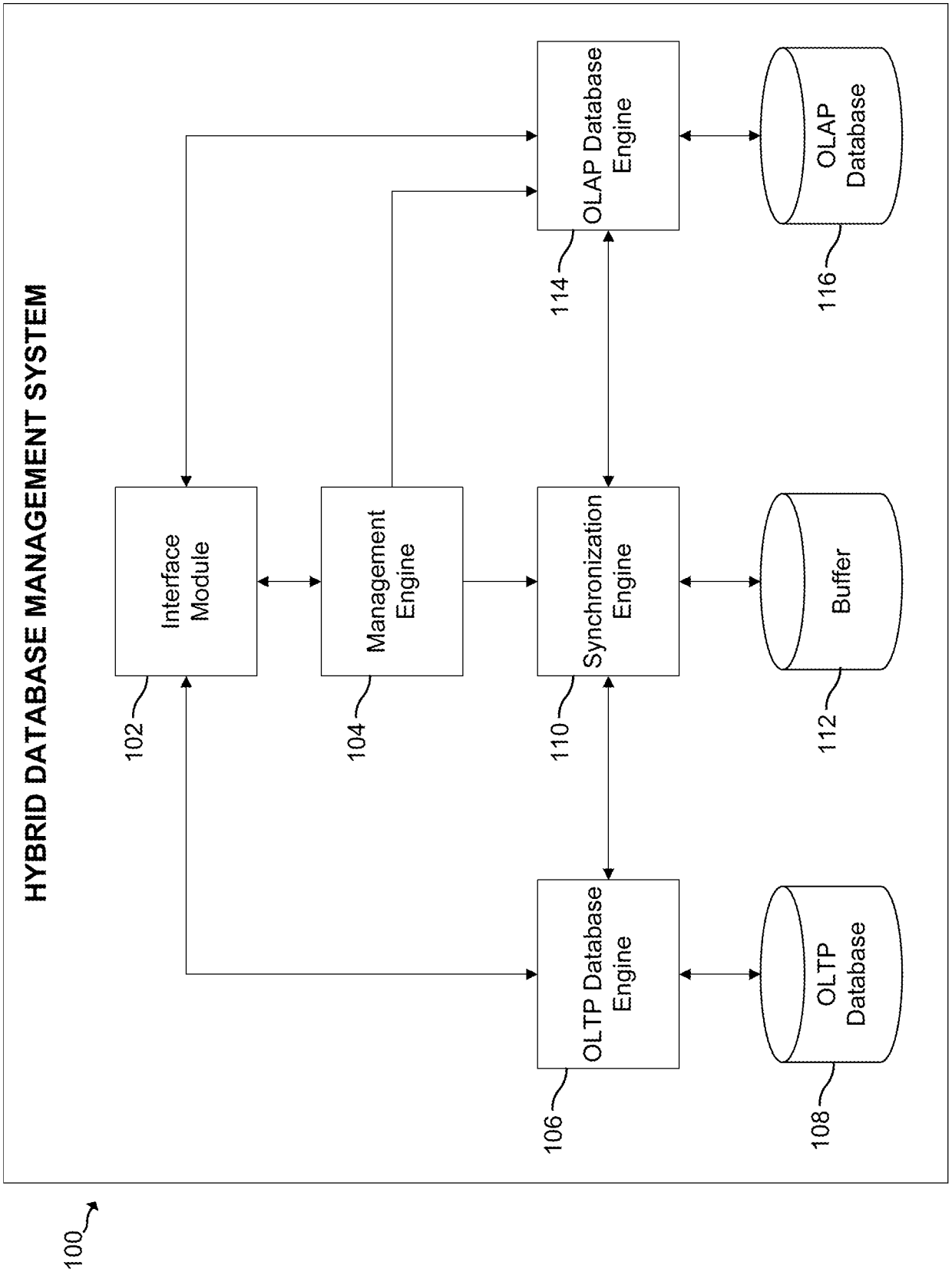
HYBRID DATABASE MANAGEMENT SYSTEM
Interface Module
Management Engine
Synchronization Engine
OLAP Database Engine
OLTP Database Engine
Buffer
OLAP Database
OLTP Database
102
104
110
112
114
116
106
108
100

FIG. 1

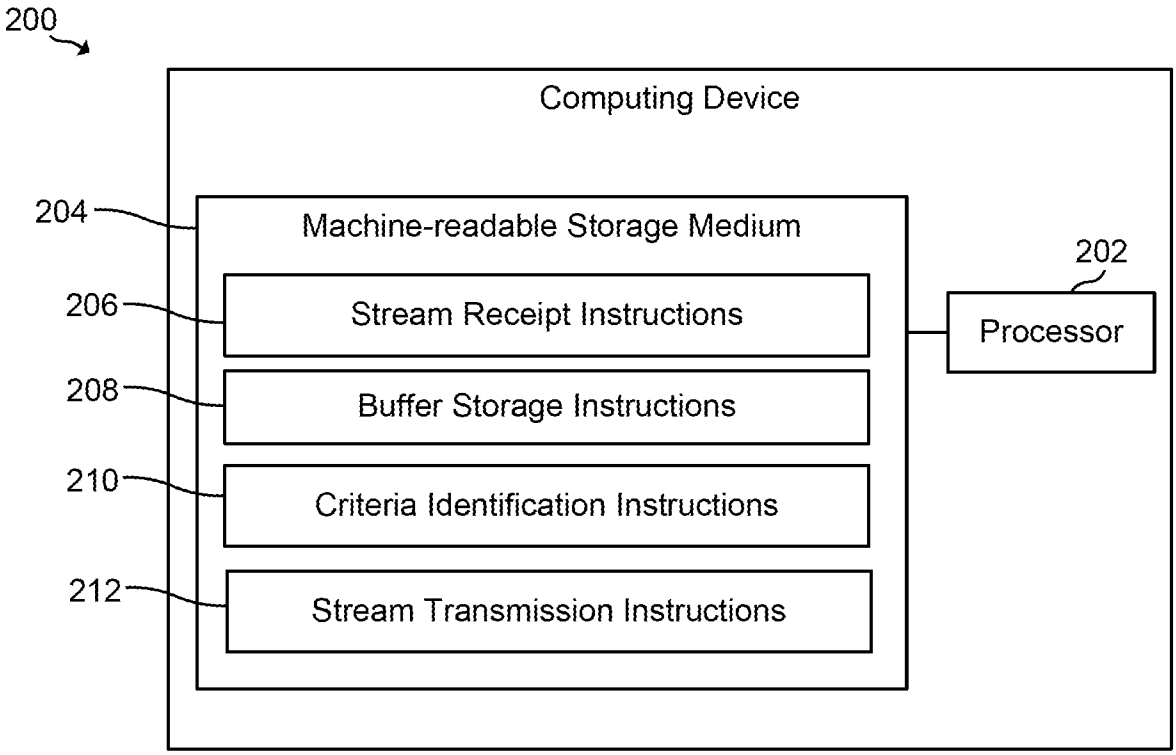
200
Computing Device
204
Machine-readable Storage Medium
206
Stream Receipt Instructions
208
Buffer Storage Instructions
210
Criteria Identification Instructions
212
Stream Transmission Instructions
202
Processor

FIG. 2

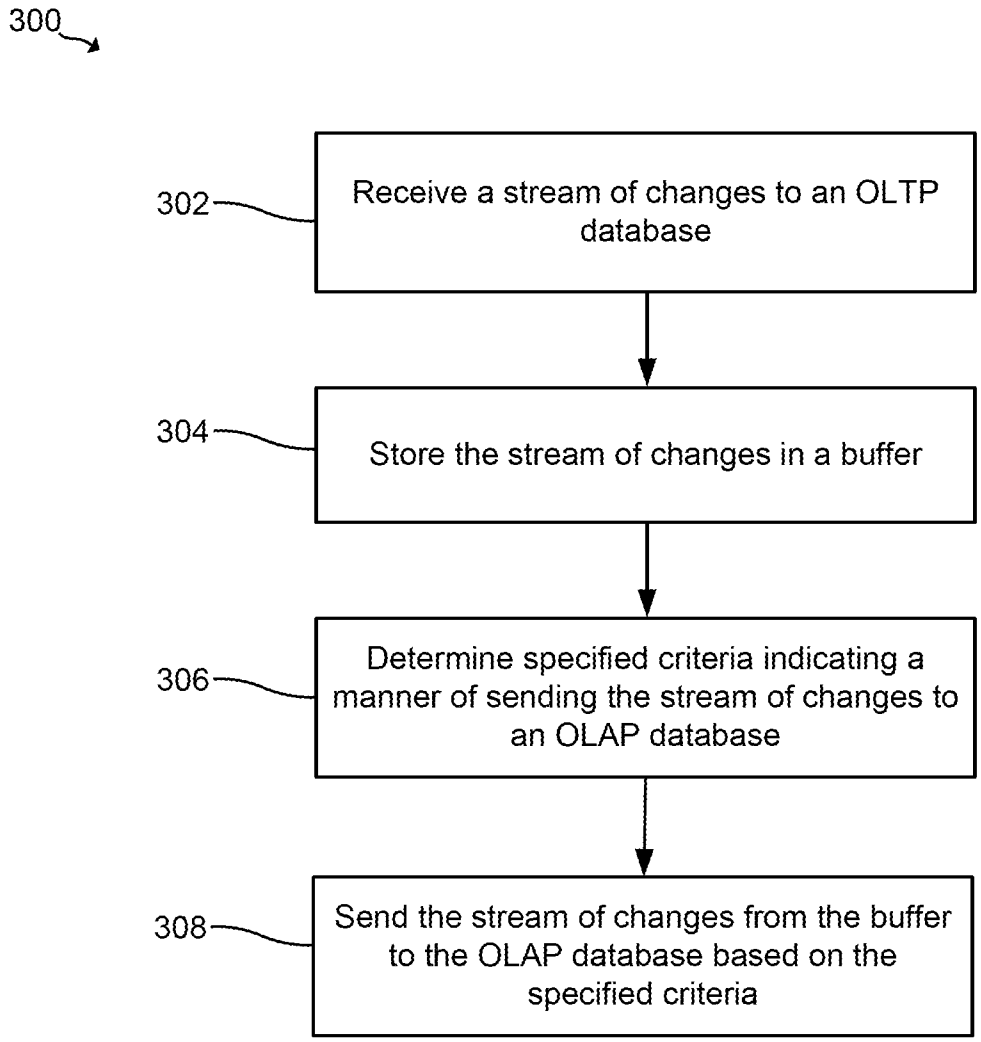
300
302
Receive a stream of changes to an OLTP database
304
Store the stream of changes in a buffer
306
Determine specified criteria indicating a manner of sending the stream of changes to an OLAP database
308
Send the stream of changes from the buffer to the OLAP database based on the specified criteria

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/067599

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/30(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 15/16; G06F 17/00; G06F 17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: OLTP, OLAP, analysis database, synchronization, period, caching, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2013-0073521 A1 (NG, RONDY et al.) 21 March 2013 See paragraphs [0054], [0057], [0087], [0089], and [0119]; claim 1; and figure 2.	1-15
Y	US 7,404,011 B2 (HANSMANN, UWE et al.) 22 July 2008 See column 5, lines 13-38; claims 1 and 9; and figure 9.	1-15
A	US 2005-0289129 A1 (SCHMITT, WINFRIED) 29 December 2005 See paragraphs [0030]-[0042] and figure 1.	1-15
A	US 2005-0278458 A1 (BERGER, ALEXANDER et al.) 15 December 2005 See paragraphs [0028]-[0033] and figure 1.	1-15

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

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 June 2015 (16.06.2015)	**16 June 2015 (16.06.2015)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701,
Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
NHO, Ji Myong
Telephone No. +82-42-481-8528

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2014/067599

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2013-0073521 A1	21/03/2013	US 2013-073520 A1	21/03/2013
		US 8959050 B2	17/02/2015
US 7404011 B2	22/07/2008	AU 2003-214262 A1	19/12/2003
		CN 100478943 C	15/04/2009
		CN 1653457 A	10/08/2005
		EP 1509858 A2	02/03/2005
		JP 2005-535947 A	24/11/2005
		TW 2004-00444 A	01/01/2004
		TW I286696 B	11/09/2007
		US 2005-0228812 A1	13/10/2005
		WO 03-102778 A2	11/12/2003
		WO 03-102778 A3	24/06/2004
US 2005-0289129 A1	29/12/2005	AT 450011 T	15/12/2009
		DE 602004024293 D1	07/01/2010
		EP 1610235 A1	28/12/2005
		EP 1610235 B1	25/11/2009
		US 7743015 B2	22/06/2010
US 2005-0278458 A1	15/12/2005	None	