(19) DANMARK

(10) DK/EP 2175384 T3

(12) Oversættelse af europæisk patentskrift

Patent- og Varemærkestyrelsen

(51) Int.Cl.: *G 06 F 16/00 (2019.01)*

(45) Oversættelsen bekendtgjort den: **2019-08-26**

(80) Dato for Den Europæiske Patentmyndigheds bekendtgørelse om meddelelse af patentet: **2019-06-12**

(86) Europæisk ansøgning nr.: **09172168.8**

(86) Europæisk indleveringsdag: **2009-10-05**

(87) Den europæiske ansøgnings publiceringsdag: **2010-04-14**

(30) Prioritet: **2008-10-10 US 249378**

(84) Designerede stater: **AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO SE SI SK SM TR**

(73) Patenthaver: **ABB Schweiz AG, Brown Boveri Strasse 6, 5400 Baden, Schweiz**

(72) Opfinder: **Luotojärvi, Mika, ABB Oy, Rantatie 12, 02730, Espoo, Finland**
Hyttinen, Riku, ABB Oy, Ilmarisentie 35, 04230, Kerava, Finland
Vihtari, Tomi, ABB Oy, Tupalantie 13.B.13, 04400, Järvenpää, Finland

(74) Fuldmægtig i Danmark: **Zacco Denmark A/S, Arne Jacobsens Allé 15, 2300 København S, Danmark**

(54) Benævnelse: **Fremgangsmåde og system til at replikere databaser**

(56) Fremdragne publikationer:
US-A1- 2003 158 868
US-B1- 6 748 381
COELHO, FABIEN: "Remote Comparison of Database Tables Technical Report A/375/CRI" TECHNICAL REPORT A/375/CRI CENTRE DE RECHERCHE EN INFORMATIQUE, 6 February 2006 (2006-02-06), pages 1-12, XP002557302 FRANCE
JAIN N ET AL: "TAPER: tiered approach for eliminating redundancy in replica synchronization" 4TH USENIX CONFERENCE ON FILE AND STORAGE TECHNOLOGIES USENIX ASSOCIATION SAN FRANCISCO, CA, USA, 2005, pages 281-294, XP002557303 ISBN: 1-931971-39-0
DEMERS A ET AL: "EPIDEMIC ALGORITHMS FOR REPLICATED DATABASE MAINTENANCE" OPERATING SYSTEMS REVIEW, ACM, NEW YORK, NY, US, vol. 22, no. 1, 1 January 1988 (1988-01-01), pages 8-32, XP000003914 ISSN: 0163-5980
Internet Archive: "Internet Archive references - Remote comparison of Database Tables - Technical Report A/375/CRI", , 12 June 2006 (2006-06-12), XP055506465, Retrieved from the Internet: URL:https://web.archive.org/web/2006060100 0000*/http://www.cri.ensmp.fr/classement/d oc/A-375.pdf [retrieved on 2018-09-12]

DESCRIPTION

Scope of the invention

[0001] The present invention relates to a method and a system for replicating databases according to the preambles of Claims 1 and 7.

Prior art

[0002] Data is often stored in databases in computer systems. When a database is replicated the same data is stored on multiple storage devices. The replication is performed frequently and it gives multiple consistent copies of the same database.

[0003] An information management system collects and manages information from several sources and distributes it to defined systems. Information may be collected from all fields of technology, business or health care, for instance.

[0004] High reliability and uninterrupted operation of the process information management system (PIMS) is needed in mission critical systems such as advanced process control of an oil refinery or the operations monitoring of a nuclear power plant. Any single point of failure in the computer systems must not interrupt the real-time operation of the process information management system or the applications running on it.

[0005] Traditionally there has been used PC clustering technology to build redundant process information management systems. Then the system and the database are running in one node at the time and in case of hardware or software failure the system is started up in the standby node. A failure always causes an undesirable break in the system operation. Relational database systems are providing several concepts of building high availability solutions and database replication functionality, but the performance of them does not meet the requirements usually set to process information management systems.

[0006] Some process historian products are able to provide redundancy for process history, but they are not capable to store relational data.

[0007] Coelho (Coelho F. "Remote Comparison of Database Tables", Technical report A/375/CRI, 6 Feb 2006, p.1-12) presents an algorithm based on operations and functions available an all relational database systems to recon ciliate remote tables by identifying inserted, updated or deleted tuples with a small amount of communication. A tree of checksums which covers the table contents is computed on each side and merged level by level to identify the differing keys.

Description of invention

[0008] The purpose of the present invention is to create a method and a system for replicating databases. In order to achieve this, the invention is characterized by the features specified in the characteristics sections of claims 1 and 7. Some other preferred embodiments of the invention have the characteristics specified in the dependent claims.

[0009] A method for replicating databases, in which method at least two databases are replicated, each database comprising of one or more tables, and each table having data and a unique tree index comprising index keys and hierarchical sums of cyclic redundancy check values calculated from the data on each tree index level. In the method logical consistency of the data between corresponding tables in the databases is maintained by comparing the sums of cyclic redundancy check values of the database tables; and if a difference between the sums of the cyclic redundancy check values is found, the tree indexes are logically divided into two sub tree indexes, and the sums of the cyclic redundancy check values of the sub tree indexes are compared to each other. The comparison and division is continued until the data causing the difference is found, and the inconsistent data is replicated between the databases.

[0010] A computer-readable medium configured with instructions that when executed by one or more processors cause carrying out a method for replicating databases, in which method at least two databases are replicated, each database comprising of one or more tables, and each table having data and a unique tree index comprising index keys and hierarchical sums of cyclic redundancy check values calculated from the data on each tree index level, in which method logical consistency of the data between corresponding tables in the databases is maintained by comparing the sums of cyclic redundancy check values of the database tables; and if a difference between the sums of the cyclic redundancy check values is found, the tree indexes are logically divided into two sub tree indexes, and the sums of the cyclic redundancy check values of the sub tree indexes are compared to each other and the comparison and division is continued until the data causing the difference is found, and the inconsistent data is replicated between the databases.

[0011] A system for replicating databases comprising: at least two databases comprising of tables having one or more tables, and each table having data and a unique tree index comprising index keys and hierarchical sums of cyclic redundancy check values calculated from the data on each tree index level, and means for maintaining logical consistency of the data between corresponding tables in the databases by comparing the sums of cyclic redundancy check values of the database tables; and means for finding a difference between the sums of the cyclic redundancy check values, means for logically dividing the tree indexes into two sub tree indexes, and means for comparing the sums of the cyclic redundancy check values of the sub tree indexes to each other and means for continuing the comparison and division until the data causing the difference is found, and means for replicating the inconsistent data is between the databases.

[0012] In the method for replicating databases an index is an ordered set of references, for example pointers, to the records or data rows in a database file or table. An index is based on one or more columns of the table. An index structure in the database is a B-tree or a B+ tree, for example. A B-tree is a data structure that maintains an ordered set of data and allows efficient operations to find, delete, insert, and browse the data.

[0013] The method and system for replicating databases is used to ensure the consistency of databases for instance in a process information management system. When one of the servers providing database services in the process information management system is down for instance due to hardware failure or a disk system upgrade, the other server or servers are running. The data supplied to the process information management system is stored to the databases of the running server or servers. The data is coming from the control and monitoring systems from a plant and comprises measured operating values or parameters from devices and calculated control values for the process devices. After the recovery of the server it is missing the data supplied to other servers during the down time. Then consistency of the databases in the servers is controlled by the method and system for replicating databases, i.e. the missing or changed data is transferred to the server which has been down and also the consistency of the servers which have been running is checked. When the replication of the databases is ready, the databases have the same stored information content and any of the databases can be in operation in the process information management system.

[0014] The method for replicating databases enables uninterrupted operation in case of any single point of failure in the computer systems and uninterrupted operation when rebuilding the system back to original one after the failure.

[0015] Other advantages are the high availability and better data access performance in case of load sharing. The high availability database provides the load sharing possibility between the redundant computers which can not be made with PC cluster technology.

[0016] The system for replicating databases does not need a transaction archive or similar that some of the relational databases are using in replication. Transaction archive is consuming storage space and decreasing performance. The copying and checking of information is a separate on-demand process.

[0017] The method for replicating databases gives the databases high availability and provides possibility to run two or more instances of the database system concurrently without any break in system operation in case of single computer failure. The system for replicating databases provides the option to build the system that contain several replicated databases in one system to provide higher level availability. The load sharing functionality provides possibility to scale the database system up to solutions which contain heavy calculations and/or very high number of concurrent users.

[0018] The system for replicating databases can be realized with low cost hardware. It enables building of the highly reliable real time process information management systems with

inexpensive PC computers.

[0019] In the system for replicating databases the hardware can be decentralized, e.g. databases can be stored in different computers, without any special hardware. The decentralization is secure from a physical point of view, e.g. against fire risk.

Figures

[0020] In the following the invention will be described in more detail with the help of certain embodiments by referring to the enclosed drawings, where

- Figure 1 is a general illustration of a system for a database replication;
- Figure 2 is a general illustration of hierarchical cyclic redundancy check values stored in a B+Tree;
- Figure 3 is a general illustration of a system for a database replication for a process information management system.

Detailed description

[0021] Figure 1 illustrates a database replication system.

[0022] In the method for replicating databases two or more databases are replicated with each other. In Fig. 1 there are three databases 1A-1C. In the present embodiment the databases 1A-1C are relational databases confirming to relational model.

[0023] Each database 1A-1C comprises of tables 2a-2i containing data. In the replication method a B-tree or a B+Tree is applied as an index structure in the data tables. A B+Tree is shown in Fig.2. It keeps data sorted and allows searches, insertions, and deletions in logarithmic amortized time. The piece of data stored in a B-tree is usually called a key. In a unique index, each key is logically unique and can occur in the B-tree in only one location.

[0024] Each table 2a-2i has a unique tree index comprising index keys and hierarchical sums of cyclic redundancy check values calculated from the data on each tree index level. A tree index is an ordered set of entries. Each entry contains a search-key value and a pointer to a specific row in the table that contains the value. An index key is a data quantity composed of one or more fields from a data row.

[0025] Figure 2 represents a part of a simple three level B+ tree implementing an index for 48 data rows. The cyclic redundancy check values ($c_{1...n}$) of each data row are stored with the corresponding index keys ($k_{1...n}$) in the leaf nodes 24. An internal node 23 contains a number

of pointers to leaf nodes. A corresponding index key and a sum of all the cyclic redundancy check values in the corresponding leaf node are stored with each pointer except the last one. The cyclic redundancy check values of the leaf node pointed by the last pointer are included in the cyclic redundancy check sum in the upper internal node which is in this example the root node 22. The sum of all cyclic redundancy check values in the whole index 21 is stored with the pointer to the root node.

[0026] A B-tree consists of node structures containing index keys, and pointers that link the nodes of the B-tree together.

[0027] The method for replicating databases is based on calculating and storing sums of cyclic redundancy check values on the data to all levels of the B or B+ tree of selected logically unique index in a database table 2a-2i. It is storing structures and providing service to check the consistency of the redundant copies of the database tables 2a-2i and correct the differences in case of found inconsistencies. The cyclic redundancy check value is calculated on each key, i.e. data row in the database table, and stored to the leaf. The cyclic redundancy check values of the keys are added together to the sum of cyclic redundancy check values on the internal node to which the leafs belong to and further the sums of cyclic redundancy check values of the internal nodes are added together to the node they belong to. Finally there is one sum of cyclic redundancy check values that is calculated from all the data in the database table.

[0028] By comparing the sum of cyclic redundancy check values of the redundant copies of the database table, it can be verified whether the copies are logically consistent with each others. Logical consistency means that the tables contain valid data. In case the sum of cyclic redundancy check values of the redundant copies differ from each others, the tree index is logically divided into two sub tree indexes, and the sums of the cyclic redundancy check values of the sub tree indexes are compared to each other and so on until the actual difference in the data is found. The inconsistent data is replicated between the databases. If a data exists in several copies of the table, the modification time of the data (row modification time) is used to decide which copy of the data is the newest one. The deleted data rows are detected with deletion log that is kept in each database.

[0029] The sum of cyclic redundancy check values comparison process described allows finding the differences of two arbitrary sized tables in O (log n) time. E.g. for two tables with 1 000 000 000 rows it would take 30 iterations to find a difference between the copies. However, as there are often many differences close to each other, using blocks of for example 64 rows instead reduces the iterations down to 24.

[0030] In B-trees, internal nodes can have a variable number of child nodes within some predefined range. When data is inserted or removed from a node, its number of child nodes changes. In order to maintain the predefined range, internal nodes may be joined or split. Because a range of child nodes is permitted, B-trees do not need rebalancing as frequently as other self-balancing search trees, but may waste some space, since nodes are not entirely full.

The lower and upper bounds on the number of child nodes are typically fixed for a particular implementation. For example, in a 2-3 B-tree, each internal node may have only 2 or 3 child nodes.

[0031] B-trees have substantial advantages over alternative implementations when node access times far exceed access times within nodes. This usually occurs when most nodes are in secondary storage such as hard drives. By maximizing the number of child nodes within each internal node, the height of the tree decreases, balancing occurs less often, and efficiency increases. Usually this value is set such that each node takes up a full disk block or an analogous size in secondary storage. While 2-3 B-trees might be useful in main memory, and are certainly easier to explain, if the node sizes are tuned to the size of a disk block, the result might be a 257-513 B-tree.

[0032] Every B-tree is of some "order n", meaning nodes contain from n to 2n keys, and nodes are thereby always at least half full of keys. Keys are kept in sorted order within each node. A corresponding list of pointers is effectively interspersed between keys to indicate where to search for a key if it isn't in the current node. A node containing k keys always also contains k+1 pointers.

[0033] Each node of a b-tree may have a variable number of keys and children. The keys are stored in non-decreasing order. Each key has an associated child that is the root of a sub-tree containing all nodes with keys less than or equal to the key but greater than the preceding key. A node also has an additional rightmost child that is the root for a sub tree containing all keys greater than any keys in the node.

[0034] A b-tree has a minimum number of allowable children for each node known as the minimization factor. If t is this minimization factor, every node must have at least t - 1 keys. Under certain circumstances, the root node is allowed to violate this property by having fewer than t - 1 keys. Every node may have at most 2t - 1 keys or, equivalently, 2t children.

[0035] Since each node tends to have a large branching factor (a large number of children), it is typically necessary to traverse relatively few nodes before locating the desired key. If access to each node requires a disk access, then a B-tree will minimize the number of disk accesses required. The minimization factor is usually chosen so that the total size of each node corresponds to a multiple of the block size of the underlying storage device. This choice simplifies and optimizes disk access. Consequently, a b-tree is an ideal data structure for situations where all data cannot reside in primary storage and accesses to secondary storage are comparatively expensive or time consuming.

[0036] Figure 3 illustrates a database replication system for a process information management system.

[0037] Process information system is an example of information management system. Process information management systems (PIMS) 30 interface the various control or other systems to

gather process data, with sampling times of a second or less or more. Process information management system provides data exchange between the application programs 31, distributed control systems (DCS) 32, programmable logic controllers (PLC) 33, laboratory information management systems (LIMS) 34, manufacturing execution systems (MES) 35, collaborative production management systems (CPM) 36, computerized maintenance management systems (CMMS) 37, Advanced Process Control applications (APC) 38, and other information technology systems around. Process information management systems provides data processing services, data acquisition and access interfaces, data visualization and analyses tools, calculation and application development tools, and integrates all significant values and information and stores them in large, efficient real time relational database.

[0038] Process information management system has usually two or more independent servers providing database services to other computer programs or computers. In this example there are two servers 1D, 1E and both contain independently working, but between each other replicated database, data production and history recording and applications. In this embodiment the databases are stored in different computers.

[0039] The real time process information management has the following characteristics, for instance: input data flow from process equipment ranging from 10 to 100000 data rows per second continuously for history recording into the database; continuous data processing, aggregating, and storing of the input data flow to aggregated history values; storage of other relational application data in the same database; 1 second response time with high data access performance to applications and users.

[0040] When one of the servers 1D of the process information management system is down for instance due to hardware failure or a disk system upgrade the other server IE is running. Real time data comprising for instance measurements and/or operating parameters of process devices is supplied then only to the database tables 2j-2l of the running server IE which are updated. After the recovery of the server 1D and continuously at run time the consistency of the databases in the servers is verified and controlled. The database replication system has an automatic consistency control and recovery. A database is a number of tables, and the consistency control and recovery are based on time stamps and checksums on each row and sums of cyclic redundancy check values in logically unique index. The row with a newer time stamp is assumed to be the correct one, and a missing row is searched on a list of deleted rows.

[0041] In case of hardware failure in one of the servers there is no break in the operation of the process information management system. Neither a disk failure causes a break to the information flow. There is the possibility to rebuild the system without interruption after the failure and the system can stand several consecutive failures of the computers without interruptions in operation of the process information management system or loss of data.

[0042] The system for replicating databases allows also a disk system upgrade without a break in the process information management system.

[0043] The method for replicating databases is advantageously performed using a computer. The programs to be used are stored in the memory of the computer or on computer readable media, which can be loaded on a computing device, for example a DVD. These computer readable media have instructions for enabling the computer to execute a method.

[0044] The invention has been described above with the help of certain embodiments. However, the description should not be considered as limiting the scope of patent protection; the embodiments of the invention may vary within the scope of the following claims.

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Non-patent literature cited in the description

- **COELHO F.**Remote Comparison of Database TablesTechnical report A/375/CRI, 2006, 1-12 [0007]

Patentkrav

1. Fremgangsmåde til at replikere databaser med hvilken fremgangsmåde mindst to databaser replikeres,
hver database omfatter en eller flere tabeller, og
hver tabel har data og et unikt træindeks i form af et B-træ eller et B+-træ, hvor det unikke træindeks på hvert træindeksniveau omfatter indeksnøgler og hierarkiske summer af cykliske redundanskontrolværdier, der beregnes ud fra dataene, hvor indeksnøglerne lagres i en ikke-aftagende rækkefølge,
hvor det unikke træindeks omfatter en flerhed af bladknuder (24), en flerhed af interne knuder (23), en rodknude (22) og en sum (21) af alle cykliske redundanskontrolværdier ($c_{1...n}$) sammen med en hægte til rodknuden (22),
hver bladknude (24) omfatter cykliske redundanskontrolværdier ($c_{1...n}$) for datarækkerne i tabellen sammen med tilsvarende indeksnøgler ($k_{1...n}$),
hver interne knude (23) omfatter en flerhed af hægter til børneknuder, og for hver hægte undtagen den sidste en sum af alle de cykliske redundanskontrolværdier ($c_{1...n}$) i den tilsvarende børneknude og en tilsvarende indeksnøgle ($k_{1...n}$), hvor de cykliske redundanskontrolværdier for børneknuden, som den sidste hægte hægter til, er inkluderet i summen af cykliske redundanskontrolværdier ($c_{1...n}$) for en associeret forældreknude på det næste højere træindeksniveau;
rodknuden (22) omfatter en flerhed af hægter til børneknuder, og for hver hægte undtagen den sidste en sum af alle de cykliske redundanskontrolværdier ($c_{1...n}$) i den tilsvarende børneknude og en tilsvarende indeksnøgle ($k_{1...n}$), hvor de cykliske redundanskontrolværdier for børneknuden, som den sidste hægte hægter til, er inkluderet i summen af cykliske redundanskontrolværdier ($c_{1...n}$);
hver interne knude (23) har et variabelt antal børneknuder indenfor et foruddefineret interval for antal af børneknuder,
hvor den logiske konsistens af data mellem tilsvarende tabeller i databasen vedligeholdes ved at:
sammenligne summerne (21) af alle cykliske redundanskontrolværdier ($c_{1...n}$) for databasens tabeller;
hvis der findes en forskel mellem summerne (21) af alle cykliske redundanskontrolværdier ($c_{1...n}$), opdeles hvert træindeks logisk i to undertræindekser, og

summen af alle cykliske redundanskontrolværdier ($c_{1...n}$) for undertræindeksene sammenlignes med hinanden;
sammenligningen og opdelingen fortsættes, til de data, der forårsager forskellen, findes, og
de inkonsistente data replikeres mellem databaserne.

2. Fremgangsmåde ifølge krav 1, **kendetegnet ved, at** databaserne lagres på forskellige computere.

3. Fremgangsmåde ifølge krav 1 eller 2, **kendetegnet ved, at** databaserne er i et informationsstyringssystem, og databaserne replikeres for at sikre konsistens af databaserne, og de replikerede databaser sættes til at betjene informationsstyringssystemet.

4. Fremgangsmåde ifølge et af kravene 1 til 3, **kendetegnet ved, at** mindst to servere lagrer databaserne, og fremgangsmåden udføres efter hardware-fejl på en server eller en disk-systemopgradering på en server.

5. Fremgangsmåde ifølge et af kravene 1 til 4, **kendetegnet ved, at** informationsstyringssystemet er et procesinformationsstyringssystem, og realtidsdata, der omfatter målinger og/eller driftsparametre for procesindretninger, tilføres til mindst en af databasetabellerne.

6. Registreringsmedium til softwareproduktdata, hvor programkode lagres, hvilken programkode, når den udføres på en computer, vil medføre, at computeren udfører fremgangsmåden ifølge krav 1.

7. System til at replikere databaser, omfattende:
mindst to databaser, hvor hver database omfatter en eller flere tabeller,
hver tabel har data og et unikt træindeks i form af et B-træ eller et B+-træ, hvor det unikke træindeks på hvert træindeksniveau omfatter indeksnøgler og hierarkiske summer af cykliske redundanskontrolværdier, der beregnes ud fra dataene, hvor indeksnøglerne lagres i en ikke-aftagende rækkefølge,
hvor det unikke træindeks omfatter en flerhed af bladknuder (24), en flerhed af interne knuder (23), en rodknude (22) og en sum (21) af alle cykliske redundanskontrolværdier ($c_{1...n}$) sammen med en hægte til rodknuden (22),

hver bladknude (24) omfatter cykliske redundanskontrolværdier ($c_{1...n}$) for dataarækkerne i tabellen sammen med tilsvarende indeksnøgler ($k_{1...n}$),
hver interne knude (23) omfatter en flerhed af hægter til børneknuder, og for hver hægte undtagen den sidste en sum af alle de cykliske redundanskontrolværdier ($c_{1...n}$) i den tilsvarende børneknude og en tilsvarende indeksnøgle ($k_{1...n}$), hvor de cykliske redundanskontrolværdier for børneknuden, som den sidste hægte hægter til, er inkluderet i summen af cykliske redundanskontrolværdier ($c_{1...n}$) for en associeret forældreknude på det næste højere træindeksniveau;
rodknuden (22) omfatter en flerhed af hægter til børneknuder, og for hver hægte undtagen den sidste en sum af alle de cykliske redundanskontrolværdier ($c_{1...n}$) i den tilsvarende børneknude og en tilsvarende indeksnøgle ($k_{1...n}$), hvor de cykliske redundanskontrolværdier for børneknuden, som den sidste hægte hægter til, er inkluderet i summen af cykliske redundanskontrolværdier ($c_{1...n}$);
hver interne knude (23) har et variabelt antal børneknuder indenfor et foruddefineret interval for antal af børneknuder,
hvor systemet yderligere omfatter:
midler til at vedligeholde logisk konsistens af data mellem tilsvarende tabeller i databaserne ved at sammenligne summerne (21) af alle cykliske redundanskontrolværdier ($c_{1...n}$) for databasetabellerne;
midler til at finde en forskel mellem summerne (21) af alle cykliske redundanskontrolværdier($c_{1...n}$);
midler til logisk opdeling af træindekserne i to undertræindekser;
midler til at sammenligne summerne af alle cykliske redundanskontrolværdier ($c_{1...n}$) for undertræindekserne med hinanden;
midler til at fortsætte sammenligningen og opdelingen, til de data, der forårsager forskellen findes; og
midler til at replikere de inkonsistente data mellem databaserne.

DRAWINGS

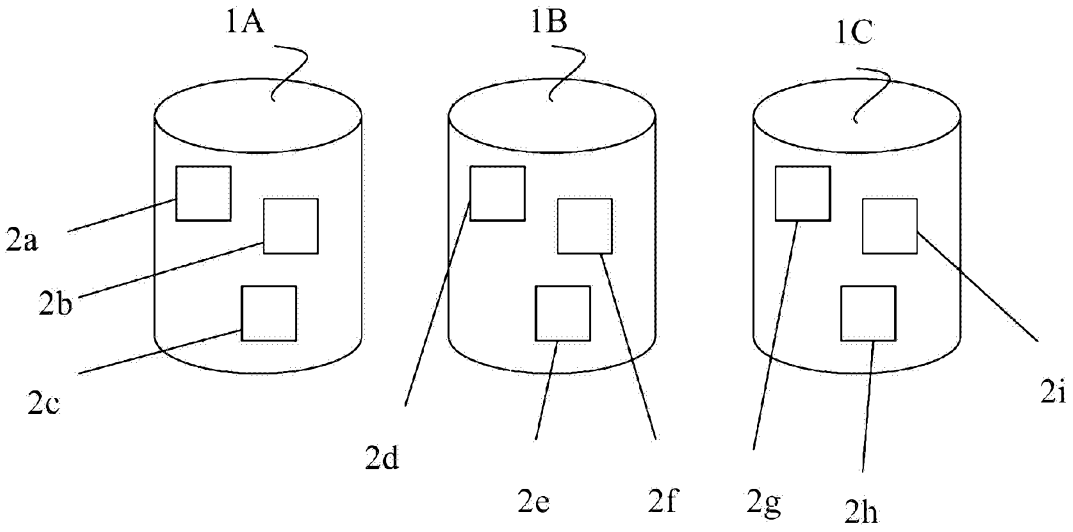

Fig. 1

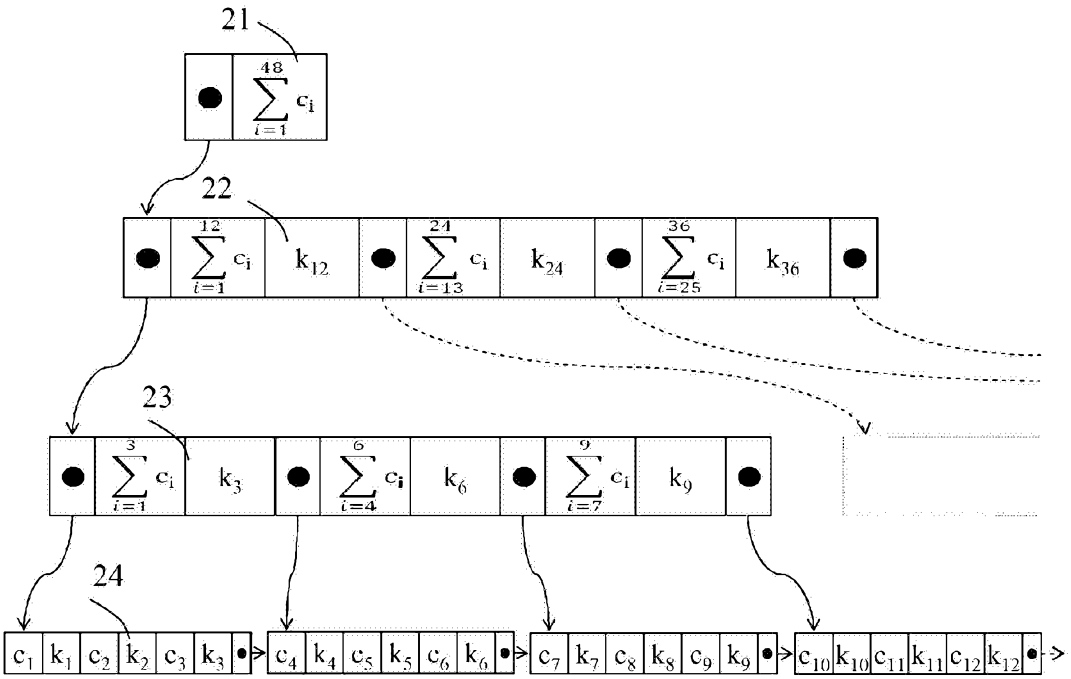
21
22
23
24
$\sum_{i=1}^{48} c_i$
$\sum_{i=1}^{12} c_i$
k_{12}
$\sum_{i=13}^{24} c_i$
k_{24}
$\sum_{i=25}^{36} c_i$
k_{36}
$\sum_{i=1}^{3} c_i$
k_3
$\sum_{i=4}^{6} c_i$
k_6
$\sum_{i=7}^{9} c_i$
k_9
c_1 k_1 c_2 k_2 c_3 k_3
c_4 k_4 c_5 k_5 c_6 k_6
c_7 k_7 c_8 k_8 c_9 k_9
c_{10} k_{10} c_{11} k_{11} c_{12} k_{12}

Fig. 2

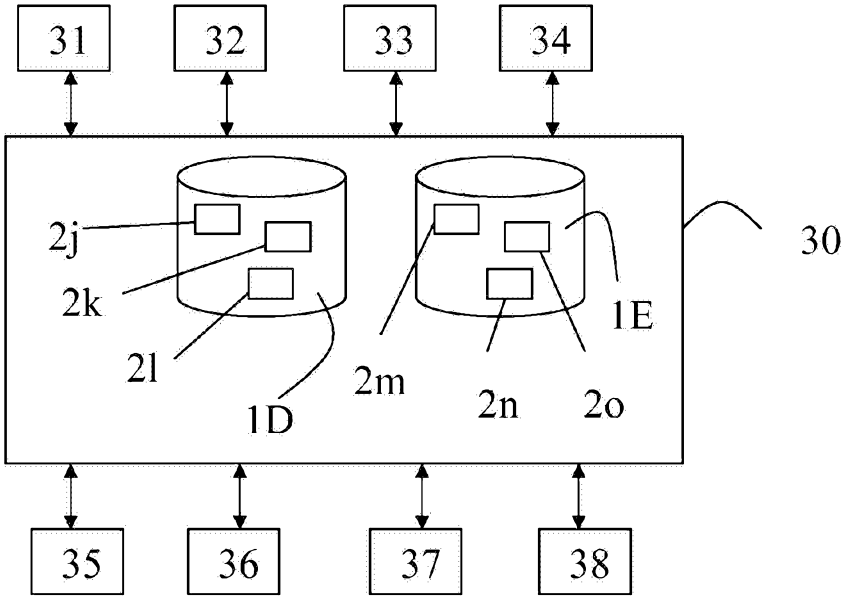

Fig. 3