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

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2522578 C 2009/06/30

(11)(21) 2 522 578

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2004/02/18
(87) Date publication PCT/PCT Publication Date: 2004/10/28
(45) Date de délivrance/Issue Date: 2009/06/30
(85) Entrée phase nationale/National Entry: 2005/10/17
(86) N° demande PCT/PCT Application No.: CN 2004/000130
(87) N° publication PCT/PCT Publication No.: 2004/092973
(30) Priorité/Priority: 2003/04/16 (CN03109877.0)

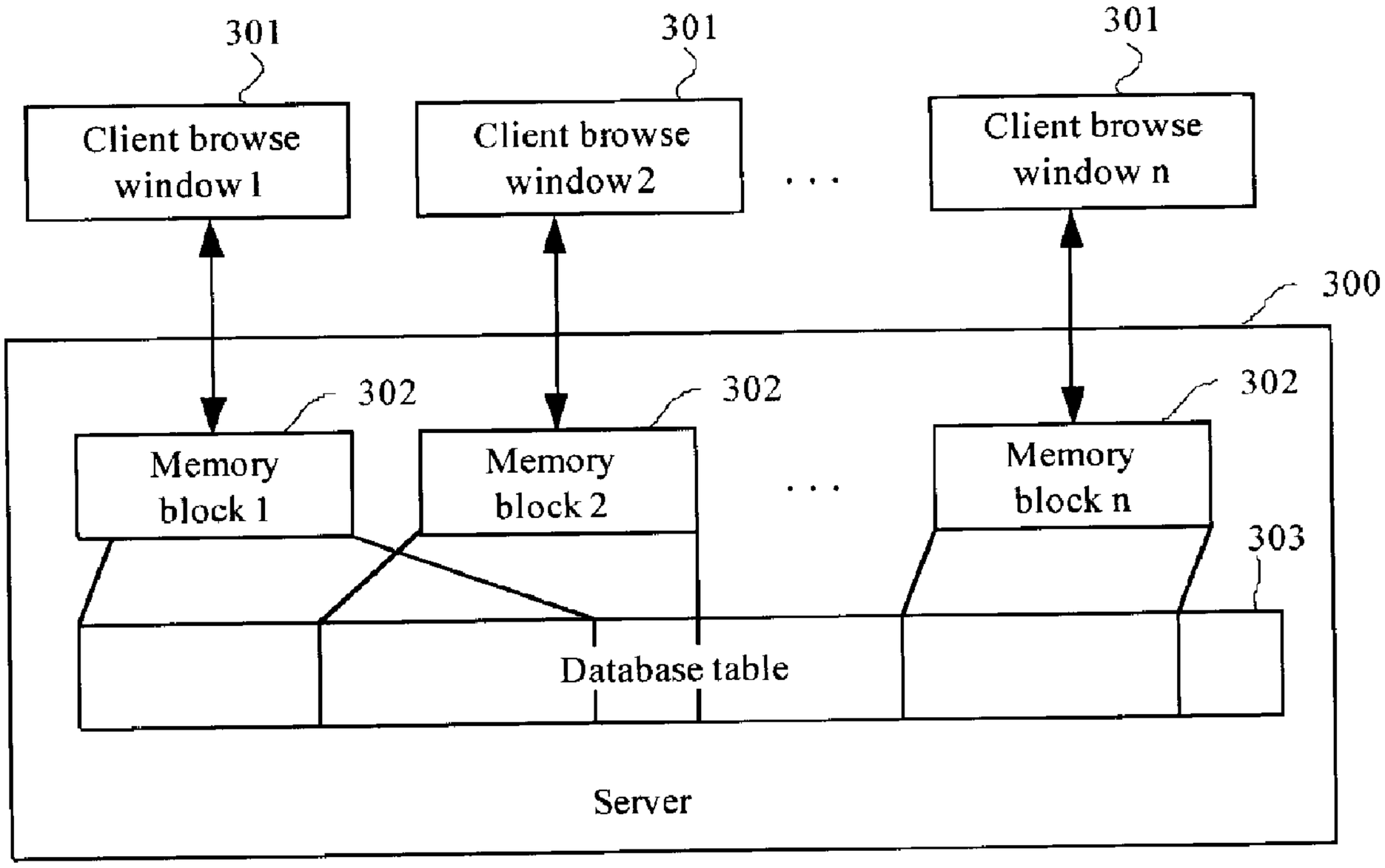
(51) Cl.Int./Int.Cl. *G06F 17/00* (2006.01), *G06F 12/00* (2006.01), *G06F 17/30* (2006.01), *G06F 3/14* (2006.01)

(72) Inventeurs/Inventors:
SONG, YI, CN;
CHEN, JIANGMING, CN;
YAO, HAO, CN;
DU, HUAKUN, CN

(73) Propriétaire/Owner:
HUAWEI TECHNOLOGIES CO., LTD., CN

(74) Agent: BERESKIN & PARR LLP/S.E.N.C.R.L.,S.R.L.

(54) Titre : TECHNIQUE PERMETTANT D'AMELIORER L'EFFICACITE DU TRAITEMENT DE DONNEES
(54) Title: A METHOD FOR IMPROVING DATA PROCESSING EFFICIENCY



(57) Abrégé/Abstract:
Disclosed is a method for improving data processing efficiency, comprising the following steps: 1) assigning each client browse window a corresponding memory block in the server, taking the amount of data to be displayed in a browse window as a data block, and saving temporarily in each memory block the N data blocks of the database that are corresponding to each browse window, respectively; 2) when the data displayed in the client browse window moves, if the data block to be displayed in the browse window and the neighboring data blocks thereof are all in the memory block, moving the data in the memory block, otherwise, saving the data block to be displayed in the browse window and the neighboring data blocks thereof in the memory block, and replacing the original data blocks in the memory block; 3) the client sending data operating commands to the server, and carrying out relevant operations on the database data; 4) after finishing the operations, updating the database with the database data in the relevant memory block. The method of the present invention can be easily implemented, and meanwhile it serves the purpose of saving system resource, increasing the speed of data processing, and improving the efficiency thereof.

Abstract

Disclosed is a method for improving data processing efficiency, comprising the following steps: 1) assigning each client browse window a corresponding memory block in the server, taking the amount of data to be displayed in a browse window as a data block, and saving temporarily in each memory block the N data blocks of the database that are corresponding to each browse window, respectively; 2) when the data displayed in the client browse window moves, if the data block to be displayed in the browse window and the neighboring data blocks thereof are all in the memory block, moving the data in the memory block, otherwise, saving the data block to be displayed in the browse window and the neighboring data blocks thereof in the memory block, and replacing the original data blocks in the memory block; 3) the client sending data operating commands to the server, and carrying out relevant operations on the database data; 4) after finishing the operations, updating the database with the database data in the relevant memory block. The method of the present invention can be easily implemented, and meanwhile it serves the purpose of saving system resource, increasing the speed of data processing, and improving the efficiency thereof.

A METHOD FOR IMPROVING DATA PROCESSING EFFICIENCY

Field of the Technology

The present invention relates to data processing techniques, and more particularly to a method for improving data processing efficiency.

Background of the Invention

With the size of database tables becoming bigger and bigger, each table can usually hold as much as tens of thousands of records, or even more. While processing the data in a database, the client configured for each data record adopts multi-window technique, Multiple Document Interface (MDI), which enables the client to open several tables simultaneously. Besides, while a database is used in a network, there are also the cases when several clients operate on one table at the same time.

In multi-window database processing, the corresponding relationship between a searching browse window and a database in the prior art is shown in Fig.1. As shown in Fig.1, the client browse window 101 is directly corresponding to the database table 102. In data processing, searching operation is important, for other operations, such as deleting, editing and sorting, in data processing can be only performed after the data are found by searching.

The basic procedure of data searching operation is shown in Fig.2, which is a schematic diagram showing the data searching procedure of the prior art. As shown in Fig.2, the client browse window 202 searches the records directly from the database 201, and the database 201 returns the found record 203 directly to the client browse window 202.

It can be seen that, ordinary data searching simply involves a direct operation on the tables in the database, therefore, when the amount of data is extraordinary large, the operation speed would be very slow. Though such a method of saving all the data in the client may reduce the times of interaction with database while searching records, large amounts of data would occupy many memory resources of the client, which is quite uneconomical. Therefore, when the amount of data is extraordinary large, the data processing efficiency of database with the method in the prior art is very low. As

the amount of data to be processed in various fields, such as communications and finance, is becoming larger and larger, it is urgent to find a method for improving data processing efficiency.

Summary of the Invention

It is the object of the present invention to provide a method for improving data processing efficiency such that data in databases could be processed in faster speed and with higher efficiency.

In order to achieve the above object, the technical scheme of the present invention is as follows:

A method for improving data processing efficiency, comprising the following steps:

1) assigning each client browse window a corresponding memory block in the server, taking the amount of data that can be displayed in a browse window as one data block, and saving temporarily in each memory block the data block of each browse window in the corresponding database and N neighboring data blocks in the database, respectively;

2) when the data displayed in the client browse window need to be moved, if the data block that needs to be displayed in the browse window and the neighboring data blocks thereof are all in the memory block, moving the data within the memory block;

if the data block that needs to be displayed in the browse window and part of its neighboring data blocks thereof are in the memory block, saving the entire data block and the neighboring data blocks thereof in the memory block and replacing the original data blocks stored in the memory block;

3) the client sending data operating commands to the server, and carrying out relevant operation on the database data within the memory block that is corresponding to the Client browse window;

4) after finishing the operations of the client browse window, closing the window, and updating the database with the database data in the memory block that is corresponding to the said browse window.

The method may further comprise: configuring an index array factory, producing an index array with a corresponding size to the requirement of the client browse window on the size of memory block, the client browse window mapping a memory

block through the corresponding index array, which takes the size of the memory block as the index.

The method may further comprise as well: when an index array is insufficient in size, requesting an index array with a larger size from the factory, and returning the old array to the factory for retrieval, the factory reserving the abandoned index array, and re-providing the said index array for use in the server when receiving a new request for being assigned a memory block.

The method may further comprise as well: when the size of the needed memory block exceeds the preset size, and the server can not find a suitable index array after receiving the request for a new memory block, deleting the least used memory block, and producing a new index array.

The said process of assigning a memory block in step 1) further comprises the following steps:

11) the client browse window sending to the server a command of assigning a memory block, which includes the name of the database table and the number of records contained in a data block;

12) after receiving the said command, the server finding out N data blocks which include the corresponding data block and the neighboring data blocks thereof in the designated database table, and saving the said N data blocks in the memory block that is corresponding to the said client browse window;

13) the server returning the records in the data block needed to be displayed in the browse window to the client.

Step 2) may further comprise:

21) Record inquiring process:

a. the client browse window sending the name of the database table and the inquiring information designated by the user to the server;

b. based on the said information, the server reading the index of the relevant record from the index table, judging whether the inquired record locates in the memory block that is corresponding to the client browse window according to the record index, if it is in the memory block, then returning the data block where the said record locates to the client and going to step c; otherwise, searching the said record in the database, returning the data block containing the inquired record to the client browse window, and going to step d;

c. judging whether the data blocks neighboring to the data block that contains the inquired record locate in the memory block that is corresponding to the client browse window, if it is in the memory block, then going to step e; otherwise, searching the neighboring data blocks in the database;

d. updating the memory block corresponding to the client browse window with the data block containing the inquired records and the neighboring data blocks thereof;

e. the client refreshing the browse window with the received inquired record;

22) Record adding process:

a. the client browse window sending to the server the name of the database table, a command of adding record and the record data to be added;

b. after receiving the command, the server inserting the designated data to the the memory block based on the received command and information, once the inserting process is successful, obtaining the index of the corresponding record, and adding the said record index into the record index table;

c. the server sending an adding record successful message, the number of the current records and a notice for refreshing the browse window to the client;

d. after receiving the said message in step c, the client refreshing data in the said browse window;

23) Record deleting process:

a. the client browse window sending the name of database table, a deleting command and the record index to the server;

b. after receiving the command, the server deleting the designated data from the memory block based on the received command and information;

c. the server sending a deleting successful message, the number of current records and a notice for refreshing the browse window to the client;

d. after receiving the said message in step c, the client refreshing the data in the current browse window;

24) Record modifying process:

a. the client browse window sending to the server the name of database table, a modifying command, the record index and the modified record data;

b. after receiving the command, the server modifying the designated data in the memory block based on the received command and information;

c, the server sending a modifying successful message and a notice for refreshing the browse window to the client;

d, after receiving the said message in step c, the client refreshing data in the current browse window.

Step 2) may further comprise: refreshing data in the client browse windows that do not initiate the operation but contain the added or deleted data during the processes of adding record, deleting record or modifying record.

Step 2) may further comprise: refreshing the sorted data in the database, wherein the refreshing operation comprises: resorting the data in the database based on the sorting rule, generating a new record index table, and instructing the client to refresh the data.

Step 2) may further comprise as wll: refreshing the sorted data in the database, wherein the refreshing operation comprises: appending the newly added record at the rear of the database table without sorting, and, when the user detects the record, sending a sorting command and the sorting rule for sorting.

The method may further comprise as well: deploying a unique Resource ID (ResID) for each record in the database, and creating a record index table in the database which takes the ResID as the record index.

The method may further comprise as well: when the pressing frequency of the Page Up or Page Down button by the user is very high or the user keeps pressing on one of those buttons for a long time, calculating the finally located page according to the times that the user has pressed the button, searching the data corresponding to the said page first in the memory block corresponding to the current browse window, if not found, then searching the said data corresponding to the current page in the database.

The method may further comprise as well: displaying the page selection button when a user has scrolled to the top or bottom of the page, searching the corresponding data in the memory block that is corresponding to the current browse window based on the user's selection, if not found, then searching the corresponding data in the database.

It can be seen from the above description of the scheme that the said method for improving data processing efficiency according to the present invention can be implemented easily. Meanwhile, it can save system resources increase the speed of data processing, and improve the efficiency thereof.

Brief Description of the Drawings

Fig.1 is a schematic diagram of the corresponding relationship between a search browse window and a database according to the prior art;

Fig.2 is a schematic diagram of the data searching procedure according to the prior art;

Fig.3 is a schematic diagram of the corresponding relationship between the client browse window and the server database table in accordance with the method of the present invention;

Fig.4 is a schematic diagram of the sliding window mechanism in accordance with method of the present invention;

Fig.5 is a schematic diagram showing the structure of the index data factory in accordance with the method of the present invention;

Fig.6 is a schematic diagram of the data searching process in accordance with the method of the present invention;

Fig.7 is a flow chart of the record inquiring process in accordance with the method of the present invention;

Fig.8-1 is a schematic diagram showing the page turning while the current data block browsed and the neighboring data blocks thereof are within the memory block;

Fig.8-2 is a schematic diagram showing the page turning while the current data block browsed and the neighboring data blocks thereof are not the memory block.

Detailed Description of the Invention

The present invention will be described in more detail hereinafter with reference to preferred embodiments and the accompanying drawings to make the object, technical scheme and advantages of the present invention more understandable.

The present invention improves the data processing efficiency mainly by the technique of window sliding.

First, assign each client browse window a memory block in the server, as shown in Fig.3, which is a schematic diagram of the corresponding relationship between the client browse window and the server database table in accordance with the method of the present invention. Assign each client browse window 301 a corresponding memory block 302 in the server 300, take the amount of data that need to be displayed in a browse window as one data block, and save temporarily in each memory block 302 the corresponding data block and the neighboring data blocks thereof in the

database 303 (totally N data blocks), respectively. The data in each memory block 302 can be overlapping or not overlapping, for example, as in figure 3, there is data overlapping between memory block 1 and memory block 2, but there is no data overlapping between memory block 1and memory block n, nor between memory block 2 and memory block n. The size of each memory block 302 in server can be determined based on specific conditions, the bigger the server memory block is, the less the times of accessing the database will be. In order to avoid large resource consumption, 3~5 data blocks are the recommended size of a memory block. In the following embodiments, the memory blocks each stores 3 data blocks: the data block corresponding to the browse window, and the previous and next neighboring data blocks thereof.

When the data displayed in the client browse window move, if both the data block that needs to be displayed in the browse window and the neighboring data blocks thereof locate in the memory block, move the data within the memory block, if not all the said data blocks locate in the memory block, save in the memory block the data block that needs to be displayed in the browse window and the neighboring data blocks thereof in the database, and replacr the original data blocks in the memory block.

In other words, the data saved in the memory block 302 move in the database table 303 according to the movement of client browse window 301, which forms a sliding window mapping the data of the database. As shown in Fig.4, which is a schematic diagram showing the sliding window mechanism in accordance with the method of the present invention, data blocks 1~3 saved in the server memory block 402 which is corresponding to the client browse window 401 are the data of data set 1 in the database table 403; if the data needed by the client browse window 401 exceed the said data blocks 1~3, and the needed data locate in data set 2, the data saved in the server memory block 402 can be changed to the data of data set 2.

Then, the client can send data operating commands to the server, and relevant operations on the database data within the memory block that is corresponding to the client browse window can be carried out.

Finally, after the operations in the client browse window are finished, close the window, and at the same time update the database with the database data in the memory block corresponding to said browse window.

In this way, when the client needs data, there is no need for the client to access the database table directly, it can send commands of data operation to the server 300 through client browse window 301, and operate on the database data in memory block 302 that is corresponding to client browse window 301. After finishing the operation, close client browse window 301, and update the database with the database data in memory block 302 that is corresponding to the browse window 301. Operating only on the memory block can greatly improve the speed and efficiency of the data processing. When the client opens several browse windows 301, each browse window 301 corresponds to one memory block 302, and there is no mutual interferences among the browse windows 301.

During the process for implementing the above scheme, a large number of memory blocks may be generated in the server, and the number of memory blocks equals to the total number of all browse windows opened in the client. Therefore, with more and more browse windows opened at the client, more memory blocks with various sizes will be generated. Meanwhile, adding or deleting data can also lead to a great deal of adding or deleting operation in the memory. In order to improve the efficiency of memory utilization and reduce the chance of generating memory fragments, in this embodiment of the present invention a memory block index array factory is established in the server for creating and reclaiming the memory blocks. The structure of the factory is shown in Fig.5. The index array factory 500 creates an index array 501 with a corresponding size according to the requirement of the browse window on the size of the memory block, and each index array 501 corresponds to one memory block 502. When the original index array in a server is not sufficient because of the newly added data, a larger index array is directly requested from the factory, and the original array is sent to the factory for reclaiming. After receiving the abandoned array, the factory will save the array rather than deleting it immediately, if a request for assigning a new memory block is received, the reclaimed array could be delivered to the server again. In order to prevent the occupied memory from increasing infinitely, when the total amount of memory blocks exceeds a certain limit, and a request is received from the server with no suitable array available, leading to the need of updating, delete the least used memory block before creating a new index array.

The advantage of this approach is that, in ideal circumstances, after a certain period of operation, the system will have enough index arrays to meet the

requirements of the server, and no more memory requests and deleting operations are needed. Moreover, in the whole processing procedure, there is no need for a series of operation of adding or deleting memory in the server only for the purpose of adding an index.

Because the operations on records in the database are extraordinarily frequent, considering the portability of database, in the present invention a unique Resource ID (ResID) is assigned for each record; with this ResID, it is possible to shield the discrepancy caused by accessing different databases with the record numbers of each database and reduce the workload for transplanting the database. Therefore, it is necessary for all the database tables to take ResID as their index and create associated index table in the database, which will increase the speed of record locating and inquiring. The ResID is assigned by the server program when a record is added. Besides the ResID set according to the present invention, the following two objects in the system can also be used to identify the records:

a. the record number in Sybase database, which is a default configuration assigned by the database;

b. the row internal address (RowID) in Oracle database, which is also a default configuration assigned by the database;

Wherein both the record number and the RowID are assigned automatically by the database, and are helpful for improving the performance of the operations. However, because the record number and the RowID belong to different databases, they must be encapsulate uniformly. In contrast, ResID is the data defined by the database itself and independent from the database system used, which cause no extra trouble in supporting a database. That is why ResID is used in the embodiment of the present invention.

The specific operations on the database data will be described in more detail hereinafter:

First of all, assign the client browse window a memory block through the foregoing process: the client browse window sending to the server a memory block assigning command, the name of the database table and the number of the records that needs to be displayed in the client browse window. After receiving the command, the server finds out three data blocks in the database: the data block displayed in the browse window, the previous and next data blocks thereof. Finally, the server returns the records in the data block to be displayed in the browse window to the client.

Thereafter, it is possible to carry out operations on the database records displayed in the browse window.

Refer to Fig.6 for the operation of record inquiring. Fig.6 is a schematic diagram of data inquiring in accordance with the method of the present invention. There are three data blocks of the database 601 stored in the server memory block 604, one of which is displayed on the client browse window 602. When the user inquires a record, the following steps will be performed: a. the client browse window 602 sending an inquiring command to the server; b. the server searching the designated data in the data blocks of memory block 604, if the inquired data is not in the memory block 604, searching the data in the database 601; after finding the data, the server sending the inquired record 603 to the client; c. the client browse window 602 displaying the result of inquiry.

As the operation of record inquiring is an important operation on the database, the process of record inquiring operation will be described in more detail hereinafter. Fig.7 is a flow chart for record inquiring in accordance with the method of the present invention, the specific process comprises the following steps:

Step 701: the client browse window sending to the server an inquiring command containing the name of the database table and the data designated by the user.

Step 702: according to the command, the server reading relevant record index (ResID) from the index table, judging according to the record index whether the inquired record locates in the memory block that is corresponding to the client browse window, if it is in the memory block, going to step 703; otherwise, jumping to step 706.

Step 703: returning to the client data block N where the said record locates.

Step 704: judging whether the neighboring data block of the inquired data block is in the memory block, if it is, going to step 709, otherwise, going to step 705.

Step 705: sending an inquiring command to the database for the said record, searching the previous and next data blocks of the data block that contains the inquired record in the database, and going to step 708.

Step 706: sending an inquiring command to the database, locating the data block N that contains the inquired record and the previous and next data blocks thereof, N+1 and N-1 in the database.

Step 707: returning data block N that contains the inquired record to the client.

Step 708: updating the memory block corresponding to the client browse window with the data block N that contains the inquired record and the previous and next data blocks thereof, N+1 and N-1.

Step 709: the client refreshing the said browse window with the received inquired records.

The record adding operation comprises the following steps: a. the client browse window sending the name of the database table, a record adding command and the record data to be added to the server; b. after receiving the said command, the server assembling SQL sentences based on the received information, inserting the designated data in the data set of the memory block, and after the inserting process is successful, obtaining the ResID corresponding to the record, adding the said ResID into the index array; c. the server sending a record adding successful message, the number of current records, and a notice for refreshing browse window to the client; d. after receiving the said message of step c, the client refreshing the data in the said browse window.

The record deleting operation comprises the following steps: a. the client browse window sending the name of the database table, a deleting command and the index (ResID) of the record to be deleted to the server; b. after receiving the command, the server assembling SQL sentences based on the received information, and deleting the designated data from the database; c. the server sending a deleting successful message, the number of current records and a notice for refreshing browse window to the client; d. after receiving the said message of step c, the client refreshing the data in the said browse window.

The record modifying operation comprises the following steps: a. the client browse window sending the name of the database table, a modifying command, the index (ResID) of the record to be modified and the modified record to the server; b. after receiving the said command, the server assembling SQL sentences based on the received information, and modifying the designated data in the database; c. the server sending a modifying successful message and a notice for refreshing browse window to the client; d. after receiving the said message in step c, the client refreshing the data in the said browse window.

During the above record adding, deleting or modifying process, while refreshing the browse window that initiates the said operation, the other browse windows that contain the added, deleted or modified records should be refreshed accordingly, and the refreshing method comprises: going through the index arrays of all the browse

windows that did not initiate the operation, if an added or a deleted record is found in a browse window in the client that did not initiate the operation, refreshing the data in the said client browse window. The appending approach can be employed for refreshing the browse window with added records, i.e. appending the ResID of the newly added record at the end of the index array. In this way, the efficiency of refreshing the index array can be greatly improved.

It is necessary for the server to refresh the index table after records are added. For a sorted database, the index thereof is special to a certain extent. When the data sets are sorted according to prescribed conditions, the index (ResID) thereof are probably not in order, therefore, it is rather difficult to insert new ResID into the index table correctly. Then, the following two methods can be adopted:

1. resorting the database with the sorting condition, producing a new index table, and then instructing the client to refresh data. This method has a rather simple processing procedure, but will increase the workload of the server, and the refreshing speed of the data set is rather slow, which, nevertheless, may not be sensed in in the client.

2. appending the newly added record at the end of the index table without sorting, which is the simplest processing method, and resorting can be conducted with re-issued conditions once the user detects the added record.

Besides, the page turning process at the client browse window is shown in Fig.8-1 and Fig.8-2. Fig.8-1 is a schematic diagram of page turning while the browsed data block and the neighboring data blocks thereof are all in the memory block; Fig.8-2 is a schematic diagram of page turning while the browsed data block and the neighboring data blocks thereof are not all in the memory block.

As shown in Fig.8-1, suppose that the number of records that need to be displayed in client browse window 811 is 100, the currently corresponding data block is data block 1 in memory block 812, and each data block has 100 records as well, while paging down, when the data to be displayed are not in data block 1, make the client browse window 811 correspond to data block 2 in memory block 812, and return the data in data block 2 to the client. In this case, no operation on the database 813 will be carried out.

Refer to Fig.8-2, suppose that the current client browse window 811 is corresponding to data block 2 in the memory block 812, while paging down, when the data to be displayed exceed data block 2, make the client browse window 811

correspond to data block 3 in memory block 812, and return the data in data block 3 to the client. In this case, as data block 4, the neighbor of data block 3, is not in memory block 812, delete the data in data block 1 that is not the neighbor of data block 3, send a data request to the database 813 simultaneously, and save the data of data block 4 in the memory block 812.

If paging up is carried out at this time, make the data block 2 in the memory block 812 correspond to the client browse window 811, return the data in the data block 2 to the client, delete the data block 4 simultaneously from the memory block 812, and add the data block 1 in front of the data block 2. The result thereof is then as shown in Fig.8-1.

In practical applications, when the user presses the Page Up or Page Down button very quickly or keeps on pressing one of these buttons for a long time, calculate the page of the final location according to the times that the user has pressed the button, search the data block that is corresponding to the page as well as the neighboring data blocks thereof in the database. When the user scrolls the window to the top or the bottom of the page, display the page selection button, and then search the corresponding data block and the neighboring data blocks thereof in the database according to the selection of the user.

It can be seen from the above embodiment, the method for improving data processing efficiency according to the present invention is easy to implement. With this method, system resources can be saved and the speed and efficiency of data processing be improved.

Claims

1. A method for improving data processing efficiency, comprising the following steps:

1) assigning each client browse window a corresponding memory block in the server, taking the amount of data that can be displayed in a browse window as one data block, and saving temporarily in each memory block the data block corresponding to each browse window and the neighboring data blocks thereof from the database (altogether N data blocks), respectively;

2) when the data displayed in the client browse window moves,

if the data block that needs to be displayed in the browse window and the neighboring data blocks thereof are all in the memory block, moving the data in the memory block;

if the data block that needs to be displayed in the browse window and part of the neighboring data blocks thereof are in the memory block, saving in the memory block the data block that needs to be displayed in the browse window and the neighboring data blocks thereof from the database and replacing the original data blocks in the memory block;

3) the client sending data operating commands to the server, and carrying out the operations on the database data in the memory block that is corresponding to the client browse window;

4) after finishing the operations for the client browse window, closing the window, and updating the database with the database data in the memory block that is corresponding to the said browse window.

2. The method for improving data processing efficiency according to claim 1, further comprising: set an index array factory, creating an index array with the corresponding size to the requirement of the client browse window on the size of the memory block, the client browse window mapping the specific memory block through the corresponding index array which takes the size of the memory block as the index.

3. The method for improving data processing efficiency according to claim 2, further comprising: when the size of an index array is insufficient, requesting an index array with a larger size from the factory, and returning the old array to the factory for reclaiming, wherein the factory reserves the abandoned index array and deliver the

index array again to the server when receiving a request for assigning a new memory block.

4. The method for improving data processing efficiency according to claim 2 or claim 3, further comprising: when the size of the needed memory block exceeds the preset size and the server can not find a suitable index array after receiving the request for assigning a new memory block, deleting the least used memory block, and producing a new index array.

5. The method for improving data processing efficiency according to claim 1, wherein the said process of assigning a memory block in step 1) further comprises:

11) the client browse window sending a memory block assigning command to the server, which includes the name of database table and the number of records contained in a data block;

12) after receiving the said command, the server finding out N data blocks which comprise the corresponding data block and the neighboring data blocks thereof in the designated database table, and then saving the said N data blocks in the memory block that is corresponding to the said client browse window;

13) the server returning the records of the data block that are needed to be displayed in the browse window to the client.

6. The method for improving data processing efficiency according to claim 1, wherein the said Step 2) further comprising:

21) record inquiring process:

a. the client browse window sending the name of the database table and the data-inquiring information designated by the user to the server;

b. based on the said information, the server reading the index of the relevant record from the index table, judging according to the said record index whether the inquired record is in the memory block corresponding to the client browse window, if it is in the memory block, then returning the data block containing the said record to the client and going to step c; otherwise, searching the said record in the database, returning the data block containing the inquired record to the client browse window, and going to step d;

c. judging whether the neighboring data blocks of the data block containing the inquired record are in the memory block that is corresponding to the client browse window, if in the memory block, then going to step e; otherwise, searching the neighboring data blocks in the database;

d. updating the memory block that is corresponding to the client browse window with the data block containing the inquired records and the neighboring data blocks thereof;

e. the client refreshing the said browse window with the received inquired record;

22) record adding process:

a. the client browse window sending the name of the database table, a record adding command and data of the record to be add to the server;

b. after receiving the command, the server inserting the designated data in the data of the memory block based on the received information, once the inserting process is successful, obtaining the index of the corresponding record, and adding the said record index into the record index table;

c. the server sending an adding record successful message, the number of the current records and a notice for refreshing the browse window to the client;

d. after receiving the message of step c, the client refreshing the data in the said browse window;

23) record deleting process:

a. the client browse window sending the name of the database table, a deleting command and the record index to the server;

b. after receiving the command, the server deleting the designated data from the memory block based on the received information;

c. the server sending a deleting successful message, the number of the current records and a notice for refreshing the browse window to the client;

d. after receiving the said message of step c, the client refreshing the data in the current browse window;

24) record modifying process:

a. the client browse window sending the name of the database table, a modifying command, the record index and the modified record data to the server;

b. after receiving the command, the server modifying the designated data in the memory block based on the received information;

c, the server sending a modifying successful message and a notice for refreshing the browse window to the client;

d, after receiving the said message of step c, the client refreshing the data in the current browse window.

7. The method for improving data processing efficiency according to claim 6, wherein the said Step 2) further comprising: refreshing the data in the client browse windows that have not initiated the operations but contain the added or deleted or modified record during the process of adding a record, deleting a record or modifying a record.

8. The method for improving data processing efficiency according to claim 6, wherein the said Step 2) further comprising: refreshing the sorted data in the database, wherein the refreshing method comprises: resorting the data in the database based on the sorting rule, generating a new record index table, and instructing the client to refresh the data.

9. The method for improving data processing efficiency according to claim 6, wherein the said Step 2) further comprising: refreshing the sorted data in the database, wherein the refreshing method comprises: appending the newly added record to the end of the database table without sorting, and when the user detects the added record, issuing a sorting command and the sorting rule so as to resort the data.

10. The method for improving data processing efficiency according to claim 6, further comprising: configuring a unique Resource ID (ResID) for each record in the database, and creating a record index table in the database taking the ResID as the record index.

11. The method for improving data processing efficiency according to claim 1, further comprising: when the user's pressing frequency of the Page Up or Page Down button is very high or the user keeps on pressing one of the buttons for a long time, calculating the page of the final location according to the times for which the user has pressed the button, searching the data corresponding to the said page first in the memory block corresponding to the current browse window, if not found, then searching the said data corresponding to the current page in the database.

12. The method for improving data processing efficiency according to claim 1, further comprising: displaying the page selection button when a user has scrolled to the top or to the bottom of a page, searching the corresponding data in the memory block that is corresponding to the current browse window based on the user's selection, if not found, then searching the corresponding data in the database.

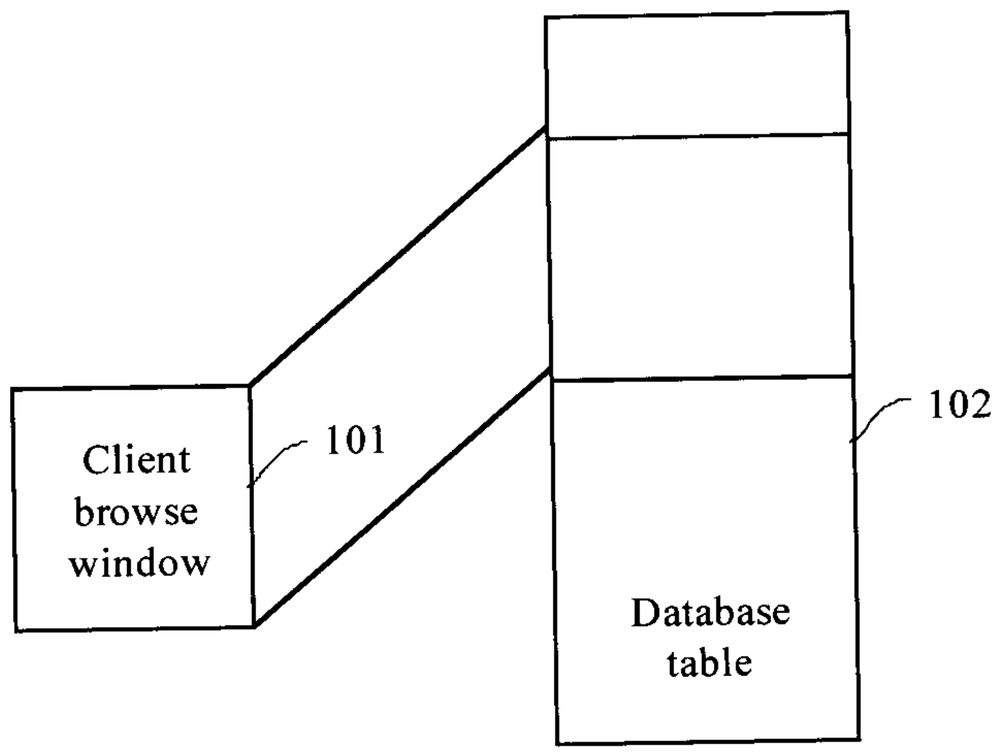

Client
browse
window
101
102
Database
table

Fig. 1

202
Client browse window
201
203
Inquired
record
Database

Fig. 2

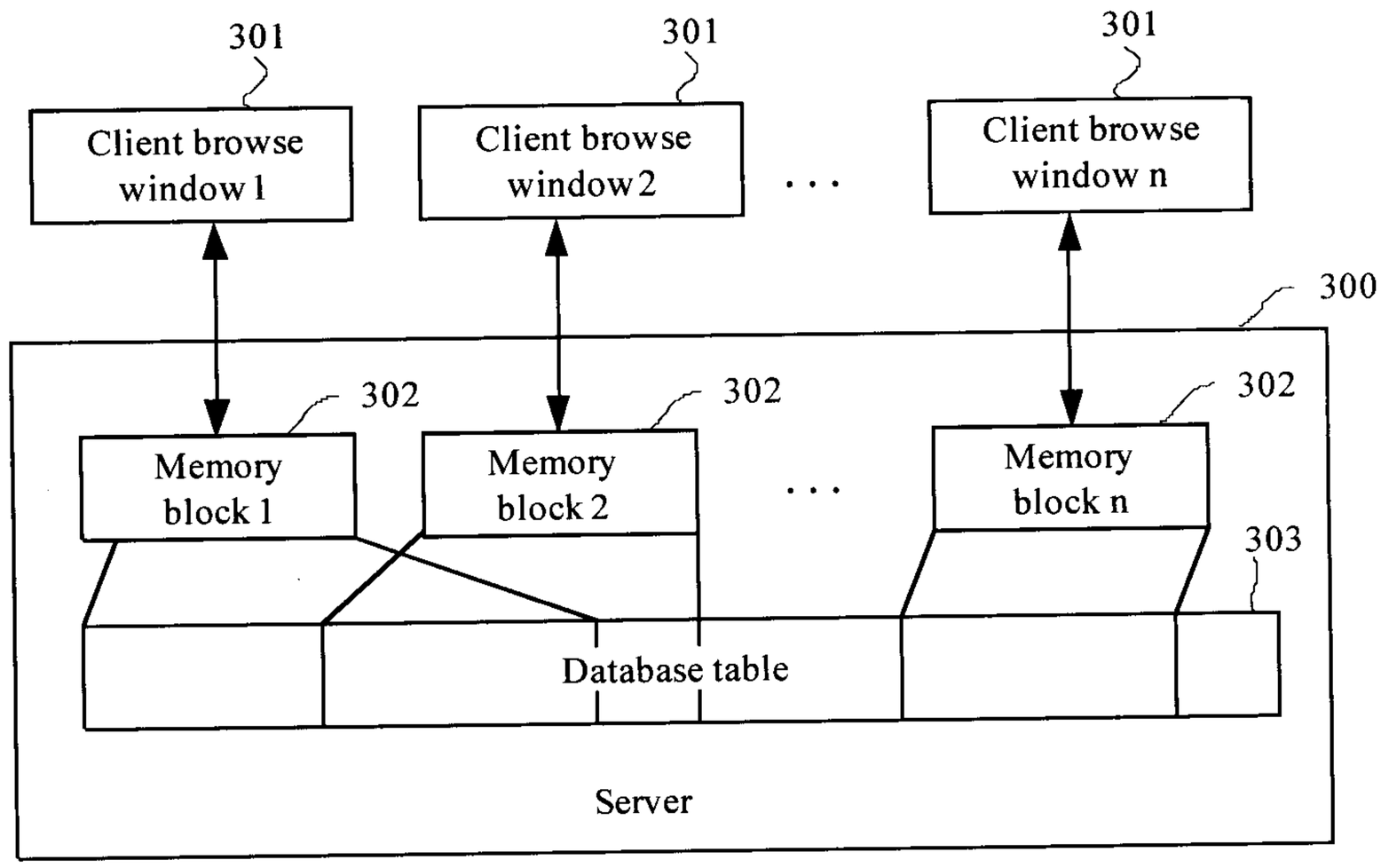

301
301
301
Client browse window 1
Client browse window 2
Client browse window n
300
302
302
302
Memory block 1
Memory block 2
Memory block n
303
Database table
Server

Fig. 4

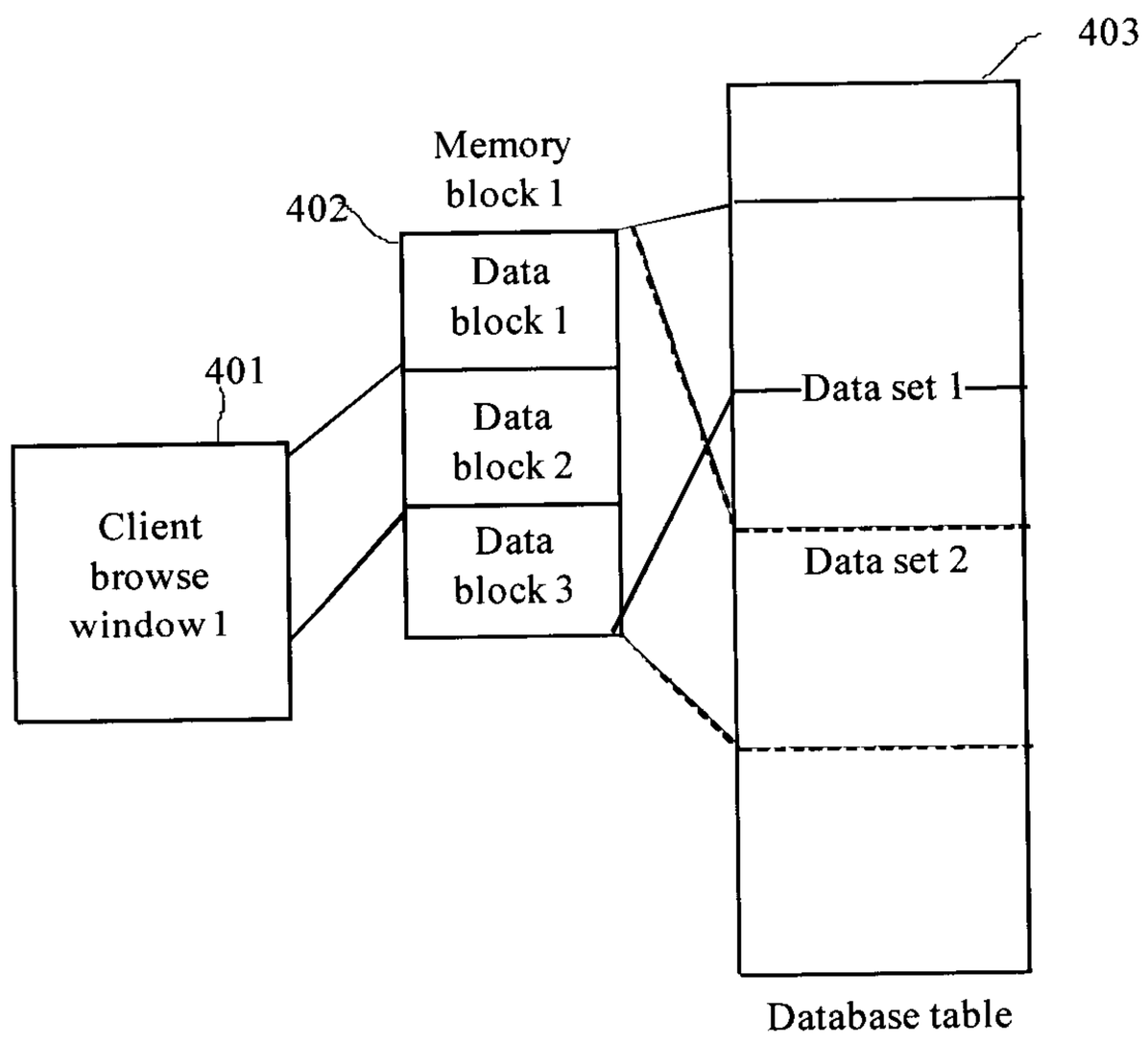

403
Memory block 1
402
Data block 1
Data block 2
Data block 3
401
Client browse window 1
Data set 1
Data set 2
Database table

Fig. 5

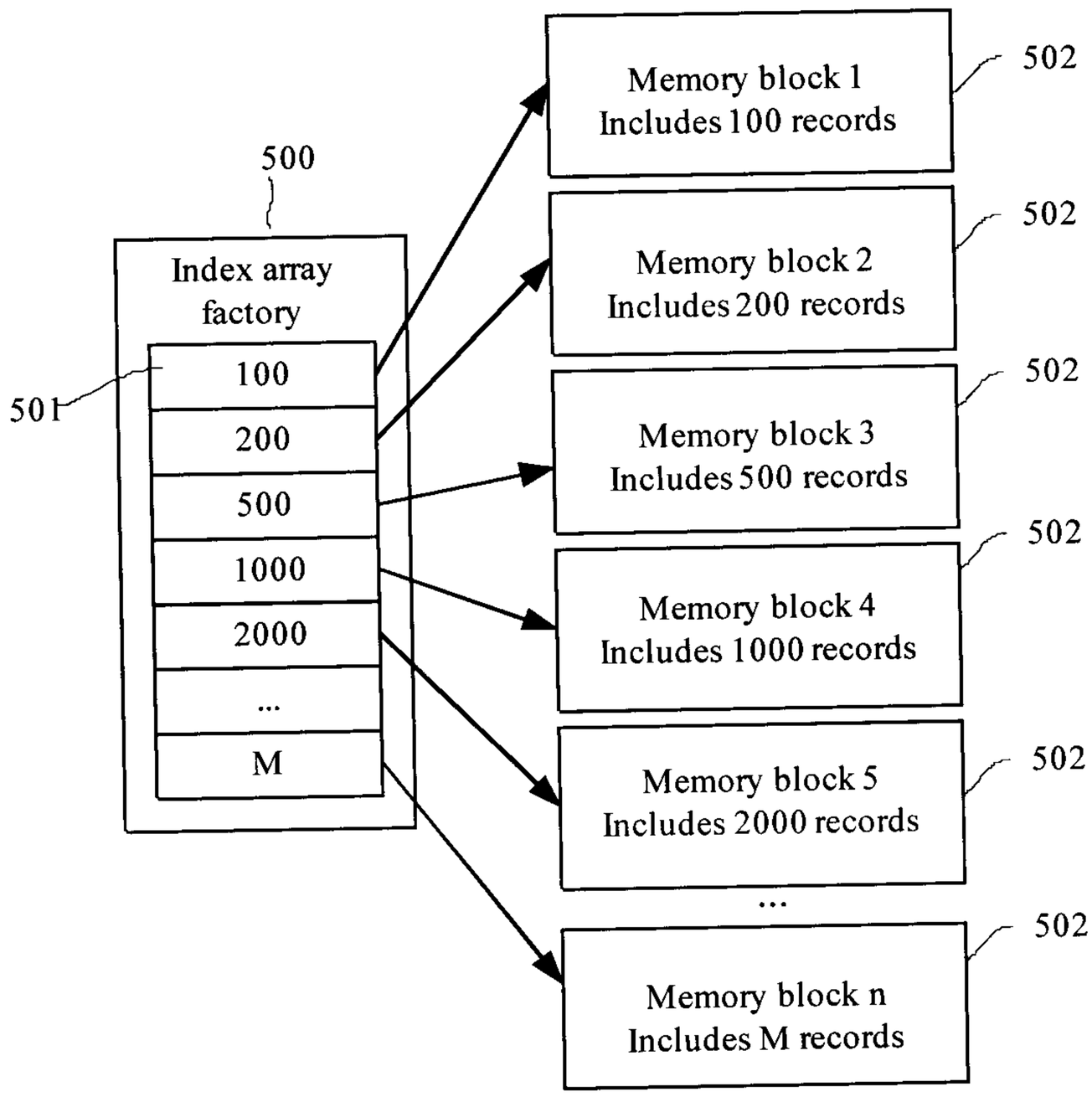

500
Index array factory
501
100
200
500
1000
2000
...
M
Memory block 1
Includes 100 records
502
Memory block 2
Includes 200 records
502
Memory block 3
Includes 500 records
502
Memory block 4
Includes 1000 records
502
Memory block 5
Includes 2000 records
502
...
Memory block n
Includes M records
502

Fig. 5

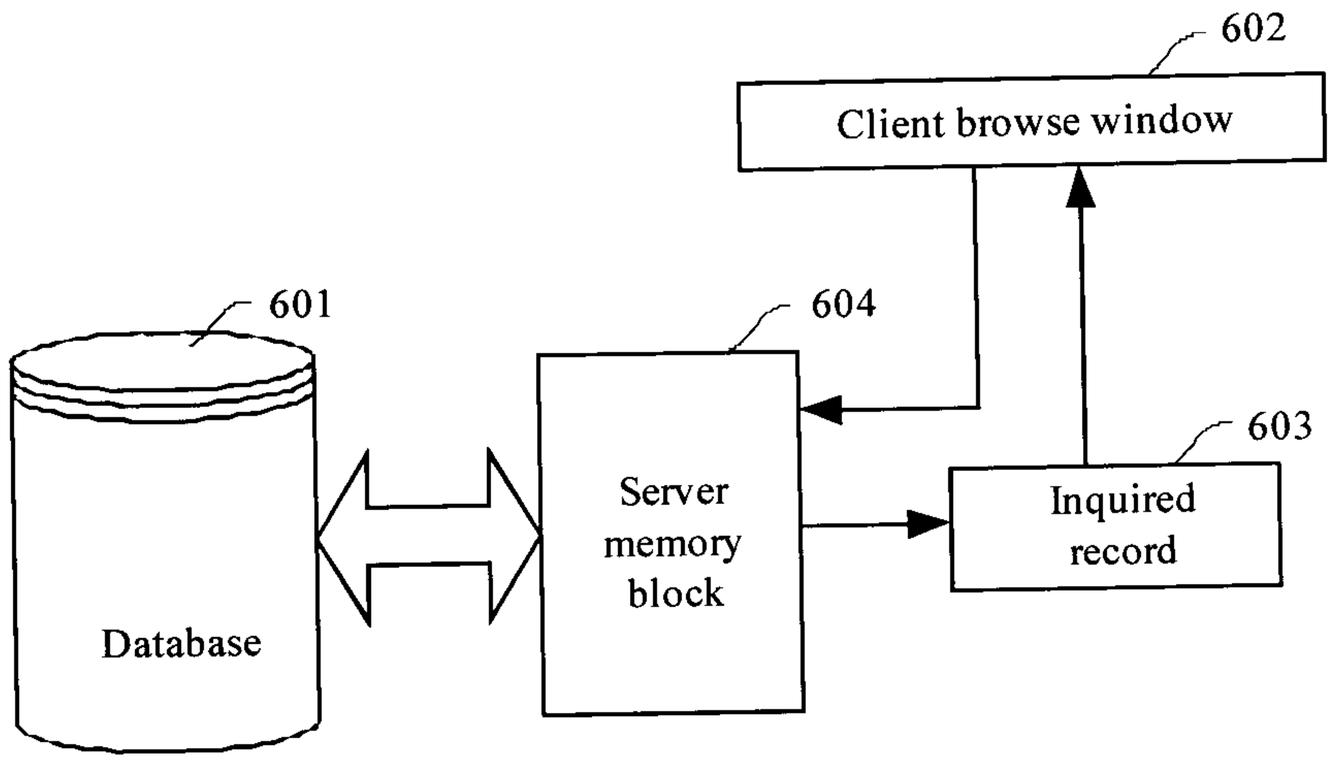

602
Client browse window
601
604
603
Server memory block
Inquired record
Database

Fig. 6

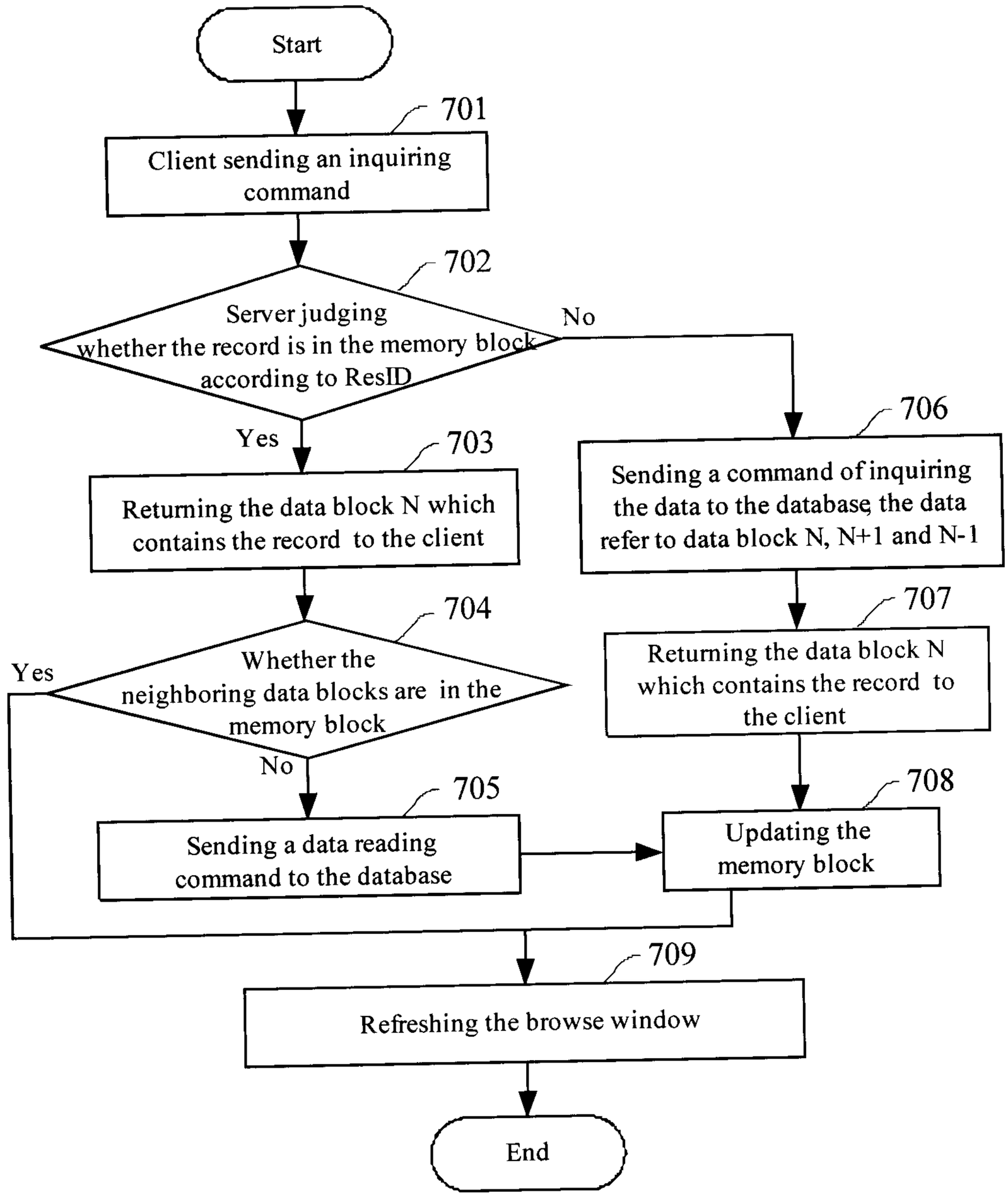

Start
701
Client sending an inquiring command
702
Server judging whether the record is in the memory block according to ResID
No
Yes
703
Returning the data block N which contains the record to the client
706
Sending a command of inquiring the data to the database, the data refer to data block N, N+1 and N-1
704
Whether the neighboring data blocks are in the memory block
Yes
No
707
Returning the data block N which contains the record to the client
705
Sending a data reading command to the database
708
Updating the memory block
709
Refreshing the browse window
End

Fig. 7

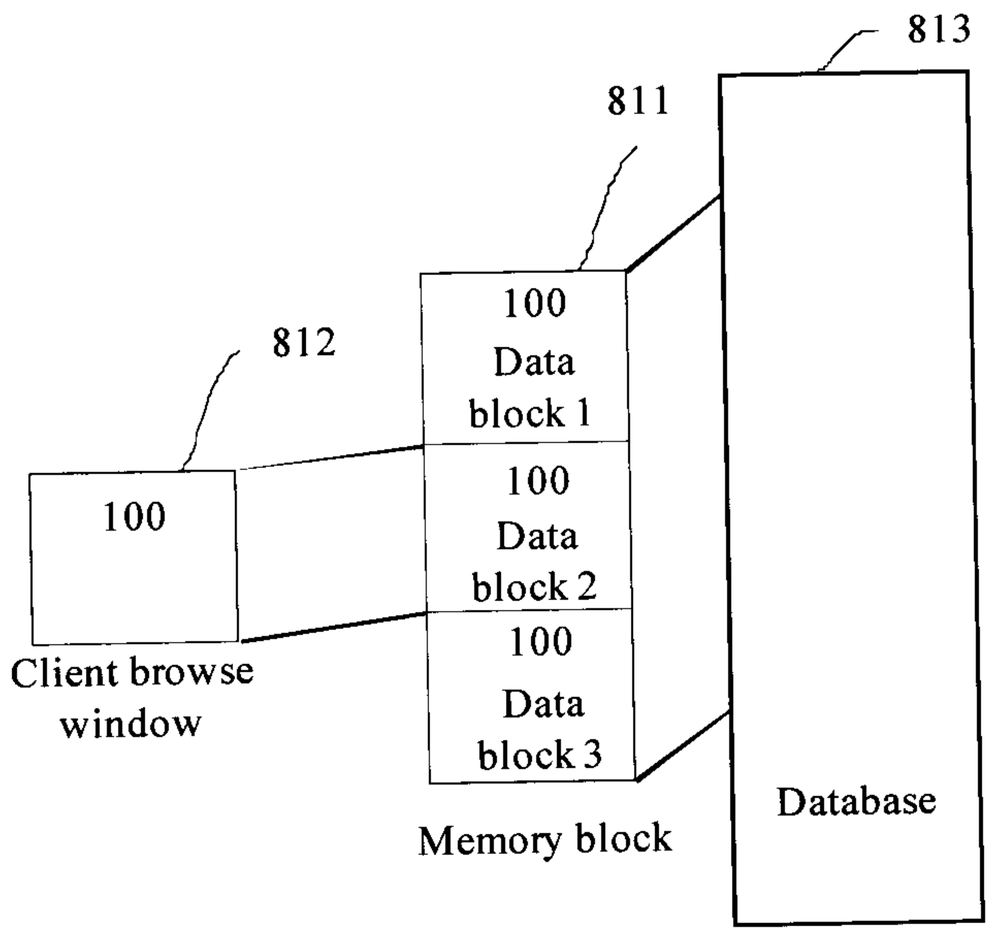

813
811
812
100
Data
block 1
100
Data
block 2
100
Data
block 3
100
Client browse
window
Memory block
Database

Fig. 8-1

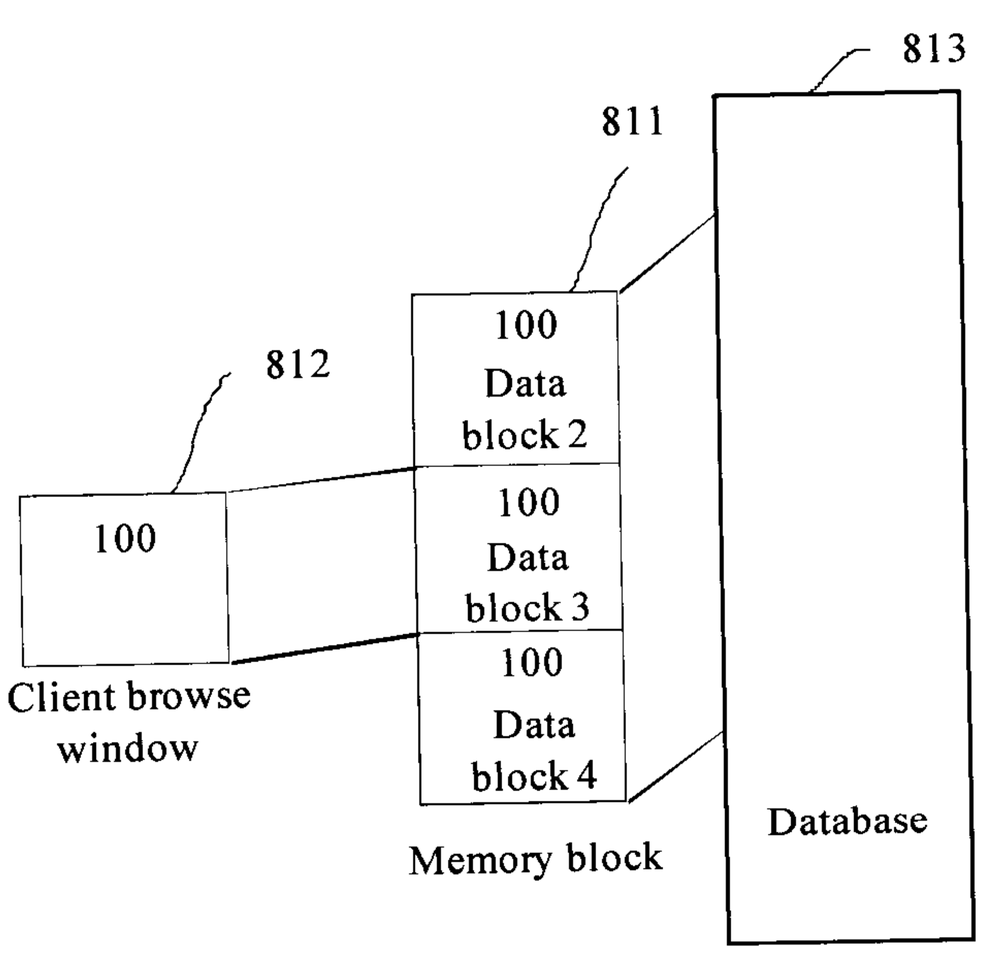
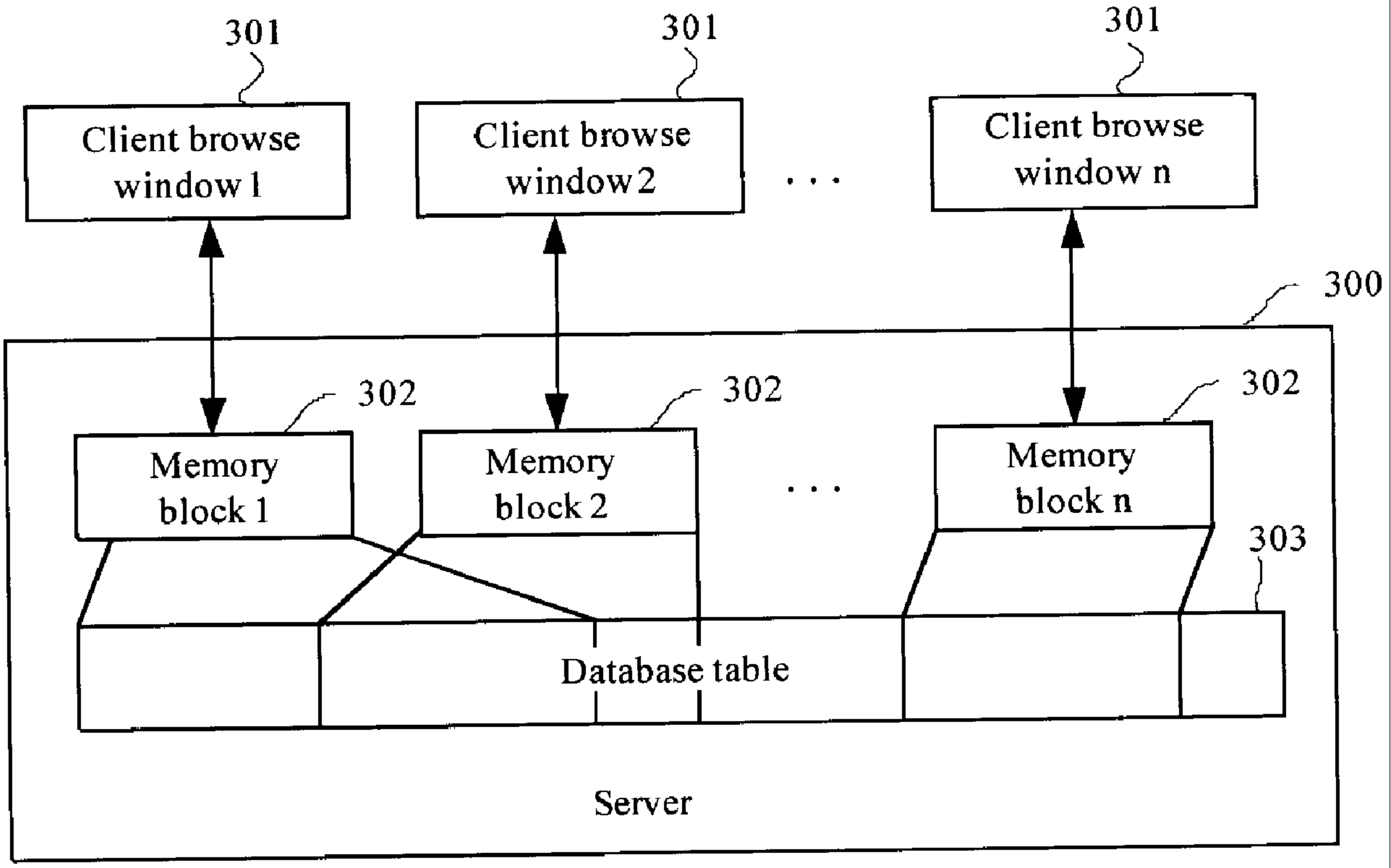

813
811
812
100
Data
block 2
100
Data
block 3
100
Data
block 4
100
Client browse
window
Memory block
Database

Fig. 8-2


301
301
301
Client browse window 1
Client browse window2
. . .
Client browse window n
300
302
302
302
Memory block 1
Memory block 2
. . .
Memory block n
303
Database table
Server