

METHOD AND APPARATUS FOR MANAGING ELECTRONIC DATA WITH
VARIABLE LENGTH ✓

5 FIELD OF THE INVENTION

The present invention relates to a method of and an apparatus for employing and storing electronic data in a computer system, and more particularly, to a process for managing and maintaining the data with variable length by method of modified-btrees (balance trees).

10 BACKGROUND OF THE INVENTION

The database system in the prior arts is usually focused on the management of electronic data with fixed record length using predefined attributes to construct index data for users to search. The multimedia database issued by "Integrate Multi-Media Database (MMDB) Based on Relational-
15 Hierarchical Model", Proceeding of National Computer Symposium, Taipei ROC, 1987, is a typical example as shown in FIG. 10.

MMDB employs a user interface 1 as a control interface for a user to communicate with the system, i.e., the user inputs an operation instruction and the system outputs the resulting data.

20 The information received by the user interface 1 is transferred to a record processing method 2, which determines the processing mode, including data storing and data fetching. After the processing mode is identified, the record processing method 2 will continuously communicate with a table processing method 3 and a tree processing method 4 in order to fetch or store
25 the record data desired. The order of the input data is not the actual order of storing the data in MMDB because the deleting process in MMDB only sets the space of the previous data to an unused status so that next new data can be stored in the unused space. Hence, there is no order issue about MMDB record.

The access and connection of the data in MMDB are implemented

by table processing using a table processing method 3 in MMDB design, which deals with all tables desired. There are 8 tables in MMDB, with which the system can connect relations for all types of data, including data file management, relation between the index tree and the record tree, relation
 5 between the attribute and the record, the result of search data, and so forth. Every process associated with the data needs one or more tables to access and execute. For example, to fetch a record of data from the database, the system first needs a TMT (Table Management Table) to determine which tree the record is and the structure of the tree and to calculate the location of the record,
 10 and a TDT (Table Type Definition Table) to deal with the arrangement format of the record data. Finally, an ADT (Attribute Definition Table) is needed to illustrate and apply the data. The table processing method 3 will construct or provide suitable information according to what the record processing method 2 requires.

15 The data in MMDB are all stored in the database with the format of trees and those data generated under the same condition are stored in the same tree such as a record tree, an index tree, or a sorting tree. Therefore, an MMDB file is a collection set of trees. While processing the tree data, the record processing method 2 obtains the desired data access rules from table
 20 processing method 3 and processes the tree data from the tree processing method 4 and then the tree processing method 4 stores the data into the corresponding tree or fetches the data to send back to the record processing method 2. There are two types of tree in MMDB, including data mode and index mode trees. For a data mode tree, the data are stored in the node of the
 25 tree such as a record tree, and the data can be fetched as soon as the location of the desired node is found; for an index mode tree, the data can only be accessed in an indirect manner with the help of a data mode tree and an index tree where the data in the node serve as an index for another tree. While MMDB constructs an index tree, the data forming the index are concurrently

stored in the index tree. To search for the desired data according to the condition, only the ID of the record which is found is stored in the find tree. In a search process, one new tree is constructed for each single condition, which is used to further construct a find tree by appropriate combination of AND or OR operations. Regarding the sort process, MMDB employs Quick Sort & Bubble Sort, and the process to construct a sort tree is the same as the process for a find tree.

MMDB uses a conventional btree as basis and a block with fixed length as the node of the tree. The length of each block is 512 bytes. The number of record data in the block is inversely proportional to the size of the record, i.e., the longer the record, the less the number of records in the block. When identifying which node is desired, the tree processing method 4 will inform the block processing method 5 which fetches the block in accordance with the node back to the tree processing method 4 to deal with and stores the data into the database.

The record data from the tree processing method 4 are fetched and analyzed by the record processing method 2, and finally sent back to the user interface 1.

There are five problems in the above-mentioned MMDB system, including:

The record length has to be fixed and variable length is not allowable.

The record is stored in random and the actual order of the record is determined only by the system that can not be changed by a user.

Due to too many tables in the system, the operation speed for single record search or conditional search is slow and it is difficult to deal with data packing because the process for rearranging the tables and the records is too complicated.

It takes too much time to accomplish a sort process due to the fact that too many tables need to be transformed.

Due to the fixed length of the block, the number of record data is reduced as the data size becomes larger so that the tree structure can not perform as well as desired.

SUMMARY OF THE INVENTION

5 The object of the present invention is to provide a method for managing the electronic data with variable length in a computer system.

 According to the invention, for the process of storing data, a record is stored into a data area and then acquire the storing address and the data length of the record. The present record set (a modified-btree) used by the
10 user and the corresponding ID of the stored data are determined. For the process of fetching data, the present record set used by the user and the corresponding ID of the fetched data are determined.

 From the desired record set and record ID, the node including the leaf of desired record is searched according to the search algorithm for a
15 modified-btree. The desired node can be found from the node buffer or the data area.

 For storing data, the record address and record length obtained are stored into a desired leaf of the desired node found. For fetching data, the address and length of the desired record are fetched from the designated leaf
20 and then the desired record data is fetched according to the address and length.

 Moreover, the present invention also provides an apparatus for managing the electronic data with variable length, which comprises a record management device, a search control device, a sort control device, a modified-btree management device, a node management device and an output device.

25 The record management device receives the requirement from the user to delete, add, or fetch the record or receive the requirement from the search control device or the sort control device, acquires the address and length of the stored record via the modified-btree device, obtains the record data from the data area, and then sends to the output device or back to the

search control device or the sort control device.

The search control device deals with the instruction of search data from the user, including single record search and conditional search. The sort control device deals with the instruction of data sorting from the user and adds
5 the designated record set to a result set (a new sort tree) according to the sort condition and the result of comparison.

The modified-btree management device manages several modified-btrees, including a system tree, find trees, and sort trees. The system tree is used to manage the data length and file address of each record in the
10 database. The find tree is used to store the result of the search process and manage the ID of the record meeting the search condition. The sort trees and find trees are of the same structure but the sort tree is used to manage the result of record sorting.

The node management device is used to fetch the desired node
15 data from a node buffer area or a data area according to the requirement of the modified-btree management device.

The present invention provides the data manager with an apparatus to manage the relation between the records in the database and all data operations. The user may combine the data in any format and with any
20 length into a piece of record and then transfer the record to the apparatus to deal with. The accessed data about the record may be placed in the system tree by the apparatus in the present invention according to the user's requirement. When the user wants to fetch a specific record data, the file address and data length of the record will be acquired from the system tree by
25 record ID (the value of order for the record) and then used to fetch the desired record data from the data area.

Besides the system tree, the apparatus also provides the find tree and sort tree which enable the user to manage the record subset. The user may fetch the record data by the system tree and transfer those IDs of the

records meeting the search condition to the find tree which will manage the records after being filtered. The user may also fetch the data about the record subset by the find tree and system tree. However, the user can perform the search process for the record subset again to construct another find tree for solving the problem of so-called partly search. Regarding the sorting process, the user just needs to designate the data for sorting (may come from the system tree, find trees, or another sort trees) and inform the apparatus of how to compare the data such that the apparatus may construct a new sort tree after being sorted.

10 Other features and advantages of the present invention will become apparent after the following description which refers to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a system block diagram of a preferred embodiment of the present invention;

FIG. 2 shows an operation flowchart of the search control device in the present invention;

FIG. 3 shows an operation flowchart of the sort control device in the present invention;

20 FIG. 4 including Figs. 4A and 4B shows an operation flowchart of the record management device in the present invention;

FIG. 5 including 5A and 5B shows an operation flowchart of the modified-btree management device in the present invention;

25 FIG. 6 shows an operation flowchart of the node management device in the present invention;

FIG. 7 shows a schematic diagram of the node structure in the present invention, wherein Fig. 7A is a schematic diagram of the node header for the modified-btree node, Fig. 7B is a schematic diagram of the leaf in the system tree, and Fig. 7C is a schematic diagram of the leaf in the find tree and

sort tree;

FIG. 8 shows a schematic diagram of searching process in the present invention;

FIG. 9 shows a schematic diagram of sorting record in the present invention, wherein Fig. 9A is a schematic diagram of the sort tree before adding
 5 sorting data, Fig. 9B is a schematic diagram of the sort tree after adding sorting data, and Fig. 9C is a schematic diagram of the data area; and

FIG. 10 shows a system block diagram in the prior arts.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

10 The present invention is an apparatus for dealing with the record data of variable length and managing searching and sorting of the data efficiently. To achieve the above object, there are three types of modified-btrees in the present invention, including a system tree, find trees, and sort trees. In FIG. 8, the tree A is a sort tree, F1 is a root node, and F2, F3, and F4
 15 are associated sub-nodes; the tree B is a system tree, S1 is a root node, and S2, S3,..., and S13 are associated sub-nodes.

The corresponding nodes of the system tree, find trees, sort trees are composed of a node header and several leaves. FIG. 7A illustrates the structure of the node header, which consists of three cells, including LeafNum,
 20 SubNum and NxtBlk. The length of each cell may be 4 bytes. LeafNum is the number of leaves in the node, SubNum is the total number of leaves in the sub-tree, and NxtBlk is the address of the root node in the sub-tree. FIG. 7B illustrates the structure of the leaf in the node of the system tree. FIG. 7C illustrates the structure of the leaf in the node of the find tree and sort tree. The
 25 length of the cell in each leaf may also be 4 bytes. Although the types of the balance trees are different, the node blocks in the above three modified-btrees are of the same size, so only one size of node block needs to be managed. The process is simplified and the speed of operation is increased. Due to the same size of the node block and the different size of the leaves in different

balance trees, the number of leaves in the node block for different balance tree is usually not the same. The leaf of the system tree in FIG. 7B is used to store the file address and record length of the record in the database. The file address and record length are stored in RecAddr and RecLen, respectively,

5 NxtBlk stores the location of the node in the next layer, i.e., the initial location of the sub-tree, and SubNum stores the total number of leaves which use NxtBlk as the root node of the sub-tree. The structure of find tree and sort tree as shown in FIG. 7C includes RecID which is the corresponding order of the record in the system tree. The node of each type of trees in the apparatus has the

10 same header which records the status of the node. The header as shown in FIG. 7A includes LeafNum that stores the number of leaves in the node, which is used at present.

FIG. 8 illustrates a schematic diagram of a search process in the present invention. Assume the system has received the search process requested by the user and the tree A is a find tree which has a root node F1. If

15 the user wants to search for the 28th record, from F1, the system will determine that the record is located at the NxtBlk sub-tree of the first leaf (because $25 < 28 < 25 + 1 + 20$), and the desired data is located at the second leaf of F3 ($28 - 25 - 1$) of which the record is related to the 791st record in the system tree from the RecID of the leaf. The tree B is also a system tree, which employs the

20 same process to search for the desired record. The record is known at the sub-tree NxtBlk of the third leaf in the node S1 (because $225 + 1 + 280 + 1 + 247 < 791 < 225 + 1 + 280 + 1 + 247 + 1 + 391$). After another search process, the record is known at the 36th record of the node S5 (because $791 - 225 - 1 - 280 - 1 - 247 - 1 = 36$). After

25 further search process downward, the desired record is known at the second leaf of the node S53. The actual record data may be found through fetching RecLen and RecAddr.

The instruction for adding or deleting the associated record will be assigned an ID, and based on the rule of balance trees and the path for

traversing the tree, the operation to add a leaf to or delete a leaf from a designated node requires modifying the number of leaves in the sub-tree (i.e., SubNum data in FIG. 7). However, a sort record process employs a method of comparison for adding new record instead of designating an ID. The method
 5 assigns a suitable condition of comparison, with which the system will compare the record data to be added to the tree with the present record data stored in the sort tree in order to find a suitable location for the record data to be inserted.

FIG. 9 is a schematic diagram of a sort record in the present invention. Fig. 9A illustrates the status of the present sort tree (the tree A) and
 10 FIG. 9C shows the record data in the data area (for convenience of explanation, the number on the header is the record ID but not file address). If a record (the data is "Japan") with ID 2300 in the present system tree is to be added to the sort tree and the comparison method of the sorting uses the order of the alphabet, the first step is to search the nearest leaf from the root node of the
 15 tree A according to binary search rule and then obtain the next layer of sub-tree from the leaf. Repeat the above process by the same binary search until the last layer of node is found so that the address of the desired leaf to be added can be obtained. For example, there are 40 leaves in the node from the node S1 of the tree A. The first record to compare is at the 20th leaf of the node S1
 20 ($40/2=20$) by binary search and the 20th leaf illustrates the 297th record. The record address and length of the 297th record are fetched from the system tree and used to fetch the desired data from the data area. In FIG. 9C, the desired data are known as "American" which is smaller than "Japan", so the search process continues from the 21st to the 40th leaf of S1. Finally, it is found that
 25 "Japan" is between the 26th leaf (the data are "Chinese") and the 27th leaf (the data are "Philippine"), so the system starts to search from the sub-tree designated by the 26th leaf, i.e., from the node S1-26, in the same manner of searching to the terminal node S125. It will be found that "Japan" is between the 2nd leaf (the data are "France") and the 3rd leaf (the data are "Malaysia")

and the 3rd leaf of S125 is the location for the record to insert. The comparison method for adding a record is similar to the aforementioned method, i.e., both methods modify the total number of leaves of the sub-tree in accordance with the path of traversing through the tree. FIG. 9C shows the result of adding
5 "Japan" to the sort tree.

FIG. 1 shows a system block diagram of one preferred embodiment in the present invention. The user sends an instruction to request an operation via a user interface 1, which includes adding, deleting, fetching, searching, and sorting records. If an instruction such as adding, deleting, or fetching the record
10 is received, the user interface 1 will directly transfer the instruction to a record management device 2. If the user interface 1 receives a sort instruction, the instruction will be transferred to a sort control device 4. When the search control device 3 receives a search instruction containing a search condition, the search control device 3 will communicate with the record management device 2 to
15 construct a new find tree and require the record management device 2 to provide all record data within the search range. Then the record management device 2 is informed of the record IDs of those records meeting the search condition and the records are added to the new find tree. This approach overcomes the problem associated with too many tables described in the prior
20 arts. The search control device 3 accesses record data by communicating with the record management device 2. A new sort tree will be constructed by a sort control device 4. Then the record management device 2 is asked to provide the desired record data and the corresponding data will be transferred to the record management device 2 record by record. Those data are required to add to the
25 new sort tree in a manner of comparison. Of course, the comparison condition from the user interface 1 is transferred to the record management device 2. The approach of executing the sort process is just like constructing a btree and comparing data without constructing complex tables. Without having to manage too many data transformation processes, it overcomes the bad performance

problem resulted from data transformation in MMDB. Besides the instruction such as adding, deleting, or fetching the record, the record management device 2 also receives those instructions from the search control device 3 and the sort control device 4.

5 The present invention employs several modified-btrees to manage database, including a system tree, find trees, and sort trees. The modified-btree management device 5 is the core device with high efficiency to manage the btrees. The system tree is used to manage the data length and file address of each record in the database. The user may easily access the record data
10 with any length if the data length and file address are obtained. Consequently, there is no longer need to fix the data length of a record as in the conventional MMDB.

When the user adds one record to the database, the data length and file address of the record will be placed in a leaf of the system tree and the location
15 of the leaf may be determined by the user or just at the end of the system tree. If a record is deleted, the system tree will delete the leaf with respect to the record. Therefore, the order of the record is just the order of the leaf in the system tree without the shortcoming of disorder in prior arts.

The find tree is used to store the result of the search process and
20 manage the ID of the record meeting the search condition. The value of the ID represents the order in the system tree. The sort tree and find tree are of the same structure but the sort tree is used to manage the result of record sorting. The modified-btree control device 5 receives the requirement from the record management device 2 to manage those btrees, including constructing the trees,
25 and adding and deleting the leaf.

Each type of tree in the present invention has a fixed length of leaves for the reason that it is necessary to fix the number of the leaves in the node so that the number of the leaves will not vary as the record length in the leaf is changed, and further extends the advantage of the btree to overcome the

disadvantage in the prior arts. The present invention sets all nodes of the trees to the same length in order to uniformly manage the node data and avoid the complexity of maintaining different length of nodes. The function of the node management device 6 is to fetch the desired node data from the node buffer area 7 or the data area 9 according to the requirement of the modified-btree management device 5. In summary, the record management device 2 can effectively deal with the file address and data length of the record having variable length by means of the modified-btree management device 5 (including adding, deleting, fetching, and modifying), and further access the record data from the data area 9 in order to output the data required by the user to the output device 8. The modified-btree management device 5 handles all the operations related to the modified-btree in the apparatus but when there is any process associated with the access of the node data, the node management device 6 will take charge of the process. The node management device 6 makes an optimal arrangement for the node used at present between the node buffer area 7 and the data area 9.

FIG. 2 is an operation flowchart of the search control device 3 in the present invention. The detailed description about the flowchart is illustrated step by step in order as follows. First, step 301 receives the search instruction, search range, and search condition from the user interface 1. Step 302 notifies the record management device 2 to construct a new search tree. Step 303 requests the record management device 2 to transfer the record data within the search range. Because the search range from the user interface 1 is just used to designate a desired tree, step 303 asks the record management device 2 to fetch all the record data about the designated tree. Step 304 fetches a record data among the record set defined by the search range. Step 305 determines if the record data meets the search condition; if not, step 307 is executed; if yes, step 306 will require the record management device 2 to add the record ID to the new search tree and then step 307 will check if all the records in the search

range are finished. If there is still any record not finished, step 304 is executed to go on managing the next piece of record data; if all the records are finished, the operation of the search control device 3 comes to an end.

FIG. 3 is an operation flowchart of sort control device in the present invention. First, step 401 receives the sort instruction, sort range, and sort condition from the user interface 1. Step 402 notifies the record management device 2 to construct a new sort tree. Step 403 requests the record management device 2 to transfer the record data within the sort range. Step 404 fetches a piece of record data among the record set defined by the sort range. Step 405 transfers the record and sort condition to the record management device 2 and requests the record management device 2 to add the record ID to the new sort tree in the manner of comparison for adding. Subsequently, step 406 will check if all the records in the sort range are finished. If there is still any record not finished, step 404 is executed to go on managing the next piece of record data; if all the records are finished, the operation of the sort control device 4 is accomplished.

FIG. 4 is an operation flowchart of record management device in the present invention. First, step 201 determines the process mode which is requested. If the record is a newly-added one, step 202 is executed to check if the tree designated for adding the record is the system tree. If not, it means the value of the input data is the record ID to be added to the search or sort tree and step 204 is executed. If yes, it means that the value of the input data is the record data and the data are then stored into the data area 9. The file address and record length can be acquired after storing the data. The file address and record length are taken as newly added leaf data and stored into the system tree. Step 204 notifies the modified-btree management device 5 to add a new leaf to the designated tree according to the record ID. Step 205 stores the data into the new leaf.

If the process of deleting record data is identified by step 201, step

206 is executed to request the modified-btree management device 5 to delete the designated leaf from the designated tree and send back the data stored in the leaf. Step 207 will determine if the designated tree is the system tree, if not, the deleting operation is accomplished; if yes, it means the data in the leaf
 5 which is deleted are the file address and data length of the record and step 208 is executed to delete the corresponding data in the data area 9.

If the processing mode is determined as comparison for adding in step 201, it means the sort process is undertaken and step 209 is executed to request the modified-btree management device 5 to provide the data in the first
 10 layer of the designated sort tree. Step 210 is then executed to use binary search in order to search the nearest leaf of the data which are to be added according to the designated comparison for adding. The modified-btree management device 5 is continuously communicated with in this step to acquire the file address and data length of the record for performing comparison
 15 process. After the nearest leaf is found, step 211 starts to check if the node is the terminal node; if not, it means the insertion point is still not found and step 212 is then executed to request the modified-btree management device 5 to provide the data in the next layer of the sub-tree which has a root on the leaf, and step 210 is executed again to repeat the above operation; if yes, it means
 20 that the insertion point is found, i.e., step 210 has found the location of the leaf, and step 213 requests the modified-btree management device 5 to add the new leaf and stores the record ID for comparison into the leaf.

If the processing mode is identified as data retrieval of record data in step 201, step 214 will be executed to inform the modified-btree management
 25 device 5 to search for the leaf data from the designated tree according to the record ID. Step 215 checks if the designated tree is the system tree; if yes, step 217 will be executed; if not, the leaf data which are found should be the record ID in the system tree, so step 216 requires the modified-btree management device 5 to search for the leaf data from the system tree with reference to the

previously found leaf data as the record ID, and then step 217 is executed. If data inquiry is requested, the leaf data which are found are taken as the file address and data length of the record and the desired record data to be inquired are fetched from the data area 9; if data modification is requested, the

5 leaf data are also taken as the file address and data length of the record, the desired record data are deleted from the data area 9, the new data are stored into the data area 9, and the file address and data length which are obtained in the above process are stored back to the leaf.

If there is a modified-btree to be deleted in step 201, step 218 will

10 notify the modified-btree management device 5 to delete the designated modified-btree. If there is a modified-btree to be added in step 201, step 219 will notify the modified-btree management device 5 to add the modified-btree.

FIG. 5 illustrates an operation flowchart of modified-btree management device in the present invention. First, step 501 determines the

15 processing mode requested. If to add a new leaf is identified, step 502 will continuously communicate with the node management device 6 in order to search for the node of the leaf to be added by traversing through the modified-btree which is designated according to the record ID obtained. Step 503 adds a new leaf to the location of the node which is found and step 504 deals with the

20 node split problem in the balance tree. Step 505 stores the data obtained to the new leaf. Step 509 will correct the information about the total number of the leaves in the sub-tree associated according to the traversing path in the modified-btree in step 502. If to delete a leaf is determined in step 501, step 506 will continuously communicate with the node management device 6 to

25 search for the node of the leaf to be deleted by traversing through the modified-btree which is designated according to the designated ID. Step 508 will deal with the problem of node combination in the modified-btree. Step 509 corrects the information about the total number of the leaves in the sub-tree associated according to the traversing path in the modified-btree in step 506. If to fetch the

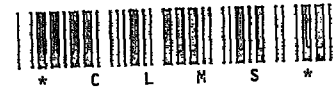
leaf data is determined in step 501, step 510 will continuously communicate with the node management device 6 to search for the node of the leaf by traversing through the modified-btree which is designated according to the designated ID. Step 511 fetches the leaf data. If to add a new tree is identified in step 501, step 512 will construct a start node of the new tree. If to delete a tree is identified in step 501, step 513 will delete all nodes of the designated tree. If to fetch the node data is identified in step 501, step 514 notifies the node management device 6 to transfer back the designated node data.

FIG. 6 shows an operation flowchart of node management device 6 in the present invention. First, step 601 checks if the node to be found is in the node buffer area 7. If it is, step 606 is executed. Otherwise, step 602 fetches the designated node from the data area 9. Step 603 checks if the node buffer area 7 is full. If it is not full, step 605 is executed. If it is full, step 604 deletes the node in LRU (least recently used) manner, which removes the node that has not been used for the longest period. If the node to be replaced is modified, the node is stored back to the data area 9. Subsequently, step 605 places the node data onto the node buffer area 7. Step 606 modifies the order value (provided for LRU) of the access process for the node and sends back the node data.

From the above-mentioned description, the apparatus of the present invention may be used to solve the access of electronic data having variable length. A modified-btree is able to connect all the records with variable length to form an intact database by a method for easily adding or deleting record and storing the order of the record. The user may construct any part of the record set by the record management device, search control device, and sort control device according to what is required. Besides, those parts of the record set can be repeatedly handled by filter, search, and sort process in order to achieve the purpose of data management and application. The node management device employs a good management algorithm to optimally adjust

and apply the resource of the node buffer area and further promote the efficiency of execution associated with the apparatus.

Although only the preferred embodiments of this invention were shown and described in the above description, it is requested that any
5 modification or combination that comes within the spirit of this invention be protected.



What is claimed is:

1. A method for managing electronic data with variable length in a computer system, including a data storing process comprising the steps of:

storing a data record into a data area, obtaining a record storing address and data length of said data record being stored, and determining a present record set and a corresponding record identifier;

searching for a desired node of a desired leaf in the records of a desired tree using the present record set and the record identifier determined in step a. according to a search algorithm of a modified balance tree;

finding the desired node of step b. from a node buffer area or a data area; and

storing the record storing address and the data length obtained in step a. into the desired leaf of step b.;

and a data fetching process comprising the steps of:

identifying a present record set and a corresponding record identifier of a requested data record;

searching for a desired node of a designated leaf in the records of a desired tree using the present record set and the record identifier identified in step e. according to said search algorithm of said modified balance tree;

finding the desired node of step f. from a node buffer area or a data area; and

fetching a requested record address and requested data length of said requested data record from said designated leaf, and then fetching said requested data record in accordance with the requested record address and the requested data length.



2. An apparatus for managing electronic data with variable length, which is used to store and employ the electronic data in a computer system having an input device, a memory device, and an output device, comprising:

- a record management device for adding, deleting and fetching a data record;

- a search control device for executing search process and managing data records which are searched;

- a sort control device for executing sort process by comparison for adding;

- a modified balance tree management device for managing a system tree, sort trees, and find trees in a modified balance tree, storing the file address and data length of a data record to the system tree, dealing with all record data in a database, and stores a record identifier of the system tree to the sort tree or the find tree as a part of the data record; and

- a node management device for managing the storage of the record data in a node buffer area and a data area.