

(11) (21) (C) **2,285,089**
(22) 1991/11/12
(43) 1993/05/13
(45) 2000/05/09
(62) 2,055,295
(22) 1991/11/12

(72) FECTEAU, JEAN GILLES, CA

(72) KLIGERMAN, EUGENE, CA

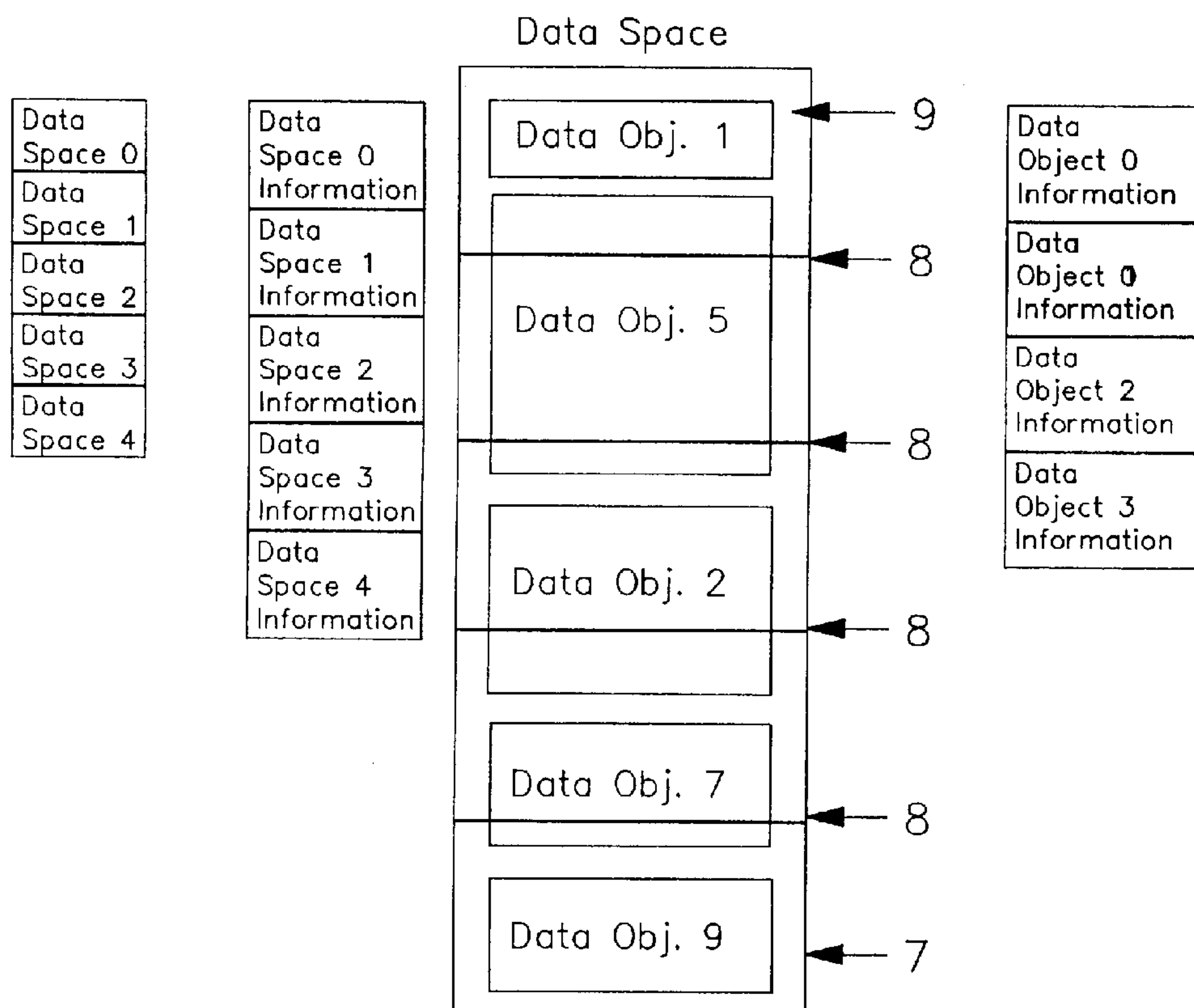
(72) KOLLAR, LUBOR, CA

(73) IBM CANADA LTD. - IBM CANADA LIMITÉE, CA

(51) Int.Cl.⁶ G06F 12/08, G06F 17/30

(54) **MISE EN CORRESPONDANCE LOGIQUE D'OBJETS
INFORMATIFS AU MOYEN D'ESPACES DE DONNEES**

(54) **LOGICAL MAPPING OF DATA OBJECTS USING DATA
SPACES**



(57) Method and means are provided for simulating a contiguous data space within a computer memory, and for placing and accessing data objects of various sizes within the simulated contiguous data space. Multiple, sub-data spaces are concatenated in such a way that each page and each sub-data space in the contiguous data space are uniquely identified. Data objects are placed in the contiguous data space and at the first reference to a page of the data object, only the segment containing the referenced page in the contiguous data space is mapped to the database storage disk. Once a data space page is mapped, the operating system can read the page into memory without requesting a disk operation from the database manager. On modifying a page, if the database disk page location is changed, the contiguous data space page is re-mapped without changing the page address in the data space. Also, modified data pages are re-written to the database storage disk in an ongoing manner set by the user, instead of at intervals set by the operating system.

CA9-91-004B

LOGICAL MAPPING OF DATA OBJECTS USING DATA SPACES**ABSTRACT**

5

10

15

Method and means are provided for simulating a contiguous data space within a computer memory, and for placing and accessing data objects of various sizes within the simulated contiguous data space. Multiple, sub-data spaces are concatenated in such a way that each page and each sub-data space in the contiguous data space are uniquely identified. Data objects are placed in the contiguous data space and at the first reference to a page of the data object, only the segment containing the referenced page in the contiguous data space is mapped to the database storage disk. Once a data space page is mapped, the operating system can read the page into memory without requesting a disk operation from the database manager. On modifying a page, if the database disk page location is changed, the contiguous data space page is re-mapped without changing the page address in the data space. Also, modified data pages are re-written to the database storage disk in an ongoing manner set by the user, instead of at intervals set by the operating system.

CA9-91-004B

1

LOGICAL MAPPING OF DATA OBJECTS USING DATA SPACES

Field of the Invention

5 The invention relates in general to method and means for mapping data objects from a simulated contiguous data space to the memory of a computer system.

The Background

10 Computerized data processing systems rely on various types of storage spaces to process and store data. For example, "main storage" is program-addressable storage from which data can be loaded directly into the registers of the central processing unit (CPU) for processing. "Auxiliary storage" is addressable storage other than main storage that can be accessed by means of input/output (I/O) channels, and includes direct access storage devices such as magnetic storage disks. "Expanded storage" is a high-speed, high-volume electronic extension of main storage that
15 is accessed synchronously in multiple-byte increments e.g. 4096 (4K) bytes, sometimes referred to as a "page".

 Data on storage disks exists in sets of fixed length records accessible in pages. The size of a page varies depending on the system in use; on some systems a page is 4096 (4K) bytes. Also, on some computers e.g. virtual machines, the operating system keeps track of disk space in blocks of
20 256 pages called segments. This is necessary for them because the hardware requires that a control block be maintained for each segment for virtual address translation. When data is to be transferred from auxiliary storage to the CPU, pages of data are transferred to a storage buffer in segments.

 In order for data to execute in the CPU, it normally should be in main storage. But main
25 storage is limited and is therefore not used to store large amounts of data permanently. On the other hand, vast amounts of data may be stored on data disks. However, accessing data from disks is slow compared to the rate at which it can be processed in main storage. To compensate for the difference in access rates, a data buffer is used. A data buffer is a portion of storage used to hold input and output data temporarily. The buffer can reside in main storage or expanded
30 storage.

 On multi-user computing systems, concurrent users time-share the resources on the computer

CA9-91-004B

2

systems through "virtual machines". In a virtual machine, which is a functional simulation of the real machine, each user is allowed to address the computer main storage as though it were real. Addressable main storage in a virtual machine is called "virtual storage". The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of real disk storage available. Data in virtual storage is mapped to real addresses when they are to be operated on by the CPU. Mapping is the establishment of correspondences between the physical storage and virtual storage locations. An address translation table is maintained by the operating system for this purpose. On virtual machines, each user's reference to a memory address is referred to as a virtual address, and each range of addressable space available to a user is called an address space. When a user references a virtual storage location, the page containing that location may be on disk or expanded storage (a "flag" in the address translation table would indicate this). When a page has to be copied to main storage, the operating system would read the page into an available real storage page location. When completed, the page translation table is updated to reflect the new page location. If no real storage space is available, the operating system frees up main storage space by "paging out" least recently used pages.

Drawings

Fig. 1 shows key information maintained in a database directory relevant to the present invention.

Fig. 2. shows an overall representation of the key elements in this invention.

Fig. 3 shows how multiple independent sub-data spaces are made to appear contiguous by using a table of pointers.

Fig. 4 shows data objects placed in specific data spaces using logical mapping.

Fig. 5 shows how a page address can be converted to a data space address and the relevant portion of the address used by this invention.

A typical database storage system is shown in Fig. 1. It comprises a directory disk (1), one or more data disks (2), and one or two log disks similar to the data disks (not as illustrated). The directory disk contains information on the mapping of the database pages from virtual storage to their real physical location and other information describing the physical configuration of the data base. The data disks stores data, while the log disks record transactions against the database. In a database system, users are assigned logical pages on disk to store data objects. A data object is a logical set of pages in the database. For example, in a relational database system,

CA9-91-004B

3

a data object may be viewed as set pages containing records of rows and columns of a table where each row is a separate record, and each column is a different data field. As shown in Fig 1, when a data object is created, entries are inserted in the database directory to indicate which data object pages contain data (3), and their physical location on disk (4). Initially, only directory space is taken, but as a user inserts data into a data object, pages are allocated on disk and the directory is updated to point to them.

In order to maintain data integrity, the database system has a mechanism to take "checkpoints" of the database at certain intervals to ensure that a consistent version of the database is saved. For example when a database page is modified, a copy of the page as of the previous checkpoint is kept unchanged, the modified version of the page is copied to disk, and the page directory is updated to point to the new location. Hence at checkpoint time the modified version of the database becomes the current copy of the database.

On virtual machines, data space pages can be mapped to disk by various techniques. For example contiguous virtual storage pages can be mapped to contiguous disk pages; this is referred to as physical mapping. Alternatively, contiguous virtual pages can be mapped to non-contiguous disk pages; this is referred to as logical mapping.

On some types of virtual machines, users can access multiple address spaces. Some of these address spaces, however, can contain only data (not computer instructions) and are referred to as data spaces. Furthermore data space pages can be mapped to disk in such a manner as to eliminate the need for the data base program manager to execute page I/O operations in order to move data between data disk and main storage. On these systems the location on disk containing a data object page is known to the operating system. When the page is referenced by a user, it is read from its disk location by the operating system without requiring a specific disk operation from the database program manager. When a page is directly accessible by the operating system, the database system operates more efficiently with less demands on CPU processing cycles.

On database systems, there is frequently a continuing need to improve the overall efficiency of the system to handle increasingly large amounts of data. In particular, there is a need for faster response to queries and data changes; efficient use of real storage; and improved efficiency in handling data objects. This in turn implies systems that optimize use of data spaces; systems that handle only those pages that are referenced rather than the whole data object; and systems that minimize database downtime at checkpoint. In the prior art, various schemes are available to use data spaces, but no method or means has been found that uses data spaces in

CA9-91-004B

4

virtual memory that handle data objects of various sizes multi-user systems. Example of prior art involving virtual storage but not addressing this deficiency include:

U.S. Patent 4,742,447, ("Method To Control I/O Accesses In Multi-tasking Virtual Memory Virtual Machine Type Data Processing System") which discloses a method for
5 accessing information in a page segmented virtual memory data processing system in which virtual machines running UNIX type operating systems are concurrently established, and in which a memory manager controls the transfer of information between primary storage and secondary storage devices in response to the occurrence of page faults.

U.S. Patent 4,843,541 ("Logical Resource Partitioning of a Data Processing System")
10 discloses a method and means for partitioning the resources in a data processing system into a plurality of logical partitions. The main storage, expanded storage, the channel and sub-channel resources of the system are assigned to the different logical partitions in the system to enable a plurality of preferred guest programming systems to run simultaneously in the different partitions.

U.S. Patent 4,843,542 ("Virtual Memory Cache for Use in Multi Processing Systems")
15 discloses a system for maintaining data consistency among distributed processors, each having its associated cache memory.

U.S Patent 4,922,415 ("Data Processing System for Converting Virtual to Real Addresses
Without Requiring Instructions from the Central Processing") discloses a method in which a
20 controller performs the translation functions for the inter-operability of the processor and memory and does so without requiring instructions from the processor.

U.S. Patent 4,961,134 ("Method For Minimizing Locking and Reading in a Segmented
Storage Space") discloses a page accessing method in a segmented table-space which eliminates
unnecessary reading and locking. The table-space comprises data pages grouped into identical-
25 sizes segments, each segment storing data for a single table. A status indicator for each data page of segment is kept in a separate segment control block stored in a space map page.

Consequently there is an unfulfilled need for means to create and use data spaces to
accommodate large data objects of different sizes, including data objects which may be much
larger than the size of a single real data space.

Also there is a unfulfilled need for means to make more efficient use of real storage by
30 mapping to data disk, only those data pages from a data space that are referenced by the CPU, rather than the entire data object.

CA9-91-004B

5

Yet another unfulfilled need is a means to efficiently save modified pages back to data disk so as to reduce the database down-time at checkpoint.

A full understanding of how the present invention addresses the above unfulfilled needs may be had by referring to the following description and claims taken in conjunction with the accompanying drawings:

General Description of Invention

The present invention provides a method for simulating a contiguous addressable data space in a computer memory, which can be a virtual memory. Multiple addressable sub-data spaces are concatenated to simulate the contiguous addressable data space. The location of each concatenated sub-dataspace is kept track of within the contiguous addressable data space. Each sub-data space is identified by an assigned unique sub-data space identifier. Each sub-data space has a known number of pages and each page has a known number of addressable storage locations. Each page is assigned a unique page identifier within the contiguous addressable data space, and each addressable location within the contiguous addressable data space can be referenced based on its unique page identifier and sub-data space identifier containing the page.

Another aspect of the invention includes means or apparatus for simulating a contiguous addressable data space in a computer memory. One embodiment can use computer programs stored or carried on suitable storage media such as magnetic or optical media. The program includes means for concatenating multiple addressable sub-data spaces to simulate contiguous addressable data space and means for keeping track of the location of each concatenated sub-dataspace within the contiguous addressable data space. An identification portion of the program provides means for identifying each of the sub-data space by an assigned unique sub-data space identifier. Each page has a unique page identifier so that each addressable location within the contiguous addressable data space can be referenced, based on its unique page identifier and sub-data space identifier containing the page.

Another aspect of the invention provides a computer system with a simulated contiguous addressable data space in computer memory. Multiple addressable sub-data spaces simulating the contiguous addressable data space are concatenated using an index containing the location of the sub-data spaces. Each sub-data space is identified by a unique sub-data space identifier, and each sub-data space has a pre-defined number of pages; and each page has a pre-defined number of addressable storage locations. The addressable storage locations are mappable to

CA9-91-004B

6

corresponding data in another storage. Each page is identified by an unique page identifier within the contiguous addressable data space and each addressable location within the contiguous addressable data space is capable of being referenced by means of a unique page identifier and the sub-data space identifier.

5

Detailed Description

An embodiment of the present invention as illustrated in Fig. 2 is incorporated in a database system that supports data objects and data space services. However it is to be understood that the invention is not restricted to the specific system described, but can be applied on a computer system that supports data spaces, data objects and data disks in the manner described herein.

The invention can be summarized as comprising of the following steps which are described in more detail in subsequent sections:

Step 1. As shown in Fig 4., on database initialization, the present invention provides for the creation, by the use of pointers (5) and a data space table (6), of a contiguous data space (7) comprising of linked, but separately identifiable, sub-data spaces (8). Each data page within the contiguous area is identifiable from an assigned page number, and each sub-area within the contiguous data space area is identifiable from an assigned sub-data space identifier.

Step 2. On first reference to a data object in the database by a user, the invention provides for enough pages in the simulated contiguous data space to be reserved to place the entire data object starting from the first available "empty" page location. In the simulated contiguous data space, the pages to be reserved start from the first available free page, and the page number is incremented sequentially to match the size of a data object. The invention provides means to identify any page of a data object within the simulated contiguous data space once the starting page number of the data object is established and its size expressed in pages is known.

Step 3. On first reference by a user to a page in the data object within the contiguous area, the invention provides for a mapping to disk only the segment of the data object containing the desired page. This partial mapping of the data object, rather than mapping the whole data object, minimizes the use of main storage because for each segment mapped, the operating system has to maintain a control block which uses up real storage. Delaying the mapping to a segment on the first reference means that a control block is only allocated if the segment is actually referenced. Therefore data space segments that are never referenced, and thus not mapped, will not require

CA9-91-004B

7

the control block.

Step 4. After pages have been modified, the invention provides for saving the modified pages at predetermined time intervals set by the user, rather than waiting for the database system to make a global save at checkpoint. This provision of the invention allows pages to be saved in an on-going manner that minimizes the database down time associated with system initiated checkpoint.

Details on the above steps are provide in the following sections.

Since a data object can be larger than the maximum size of the real data space available to a user, the first step as provided by this invention and shown in Fig 3. is to create a large apparent contiguous data space (7) by linking separately identifiable fixed-sized sub-data spaces (8). This may done advantageously by the following steps:

i) Logically link multiple fixed-sized sub-data spaces using an array of pointers, and maintain a data space table for this purpose. See Fig 3.

ii) For each page in the simulated contiguous data space, assign a unique page number as provided for by this invention.

iii) For each page in the simulated contiguous data space, establish the page number as provided for by the present invention.

In order to keep track of the identity of each data object and sub-data space within the simulated contiguous data space, a table of pointers is set up to identify the start and end of each data space and sub-data space.

Within the simulated contiguous data space, any page number can be located as provided by the present invention. For example, if the size of each constituent data space is 524,288 pages, and the page being referenced is page number 1,217,436, then we can calculate that the referenced page is 168,860 in the third sub-data space. After the page has been identified, it can be mapped to the correct real storage page.

In pseudo code format the procedure to determine the page address as a function of its location on a data space can expressed as follows:

If the Starting_page_number of the data object(n), is greater or equal to 0 (i.e. the data object was placed in the simulated contiguous data space, then do;

$$\text{Data\_space\_number} = [(\text{Starting\_page\_number of data object},(n)) + m] / [\text{the size of 1 data space expressed in pages}]$$

CA9-91-004B

8

Similarly the page number within the contiguous area can be determined as follows:

Page address = [(Starting_page_number of data object,
 (n)) + m] // [(The size of 1 data space
 5 expressed in pages) * (the size of 1 page expressed in bytes)]
 end;

10 In the above pseudo code the Starting_page_number is the location of the first page of the data object in the simulated contiguous data space; the symbol (/) represent an integer division; the symbol (//) represents the remainder of an integer division; (*) is the multiplication operator; and (n) is the identity of the data object; and (m) is the page number within the data object containing the required data.

 If the data object Starting_page_number is not known, it can be determined as follows:

15 i) On database initialization, an array named "Starting_page_number" is defined to contain the starting page location of each data object to be placed in the simulated contiguous data space. The dimension of the array is the same as the number of data objects in the database. The array is initialized to negative values to indicate that the data objects have not yet been placed in the simulated contiguous data space.

20 ii) A variable "Next_available_page" is defined and initialized to 0 in a counter to indicate that this is the beginning page of the empty simulated contiguous data space.

 iii) At the first user reference to a data object, the next available set of pages in the simulated contiguous data space is reserved for the referenced data object, by setting the Starting_page_number of the said data object to the Next_available_data space page number taken from the counter.

25 iv) The Next_available_page counter is then increased by the size of the data object (in pages) rounded up to the next segment so as to start all data objects on a segment boundary. In this invention it is convenient to start the data object on a new page segment boundary, but this is not an essential requirement if the systems does not handle pages in segments.

30 With the above procedure, any address within the simulated contiguous data space can be determined once the location of the object is known. Segments of the data object can then be mapped to its disk using information in the database directory.

 The same result can be obtained by shifting bits. For example, on a 32-bit computer

CA9-91-004B

9

architecture, bits 0 to 12 can be used to represent the data space number, and bits 13 to 31 shifted left 12 positions (equivalent to multiplication by 4096) as the address in the data space. This alternative procedure is illustrated in Fig. 5.

5 Once the simulated contiguous data space is created, a data object can be placed in the data space as shown in Fig 3. For example, data object number 1 (9) consisting of 102,528 pages is placed in pages 0 to 102,527 in sub-data space 1 (10). This may be followed by another data object number 5 (11), say, consisting of 1,024,000 pages placed from pages 102,656 (which marks the beginning of a new segment in data space 1) to pages spanning sub-data spaces 2 (8) and 3 (8). In this example, data object 1 would therefore be placed in the first 400 segments of
10 the simulated contiguous data space, while data object 5 would be placed from the first page of segment number 401 starting from page number 102,565.

Similarly, since the Starting_page_number for each data object is known, any request for a data page can be converted to a page number within the simulated contiguous data space. For example if data object(n) starts at page 1,126,656 and we request data page 90,780 of data object
15 (n), we can calculate the page number to be 1,217,436 or page 168,860 located in the third sub-data space.

An important benefit in allowing a data object to start on a segment boundary instead of at the beginning of data space is the ease in supporting both very large data objects (spanning more than one data space) and very small data objects (spanning much less than the size of one
20 data space). This invention eliminates the need to have one contiguous data space for each data object, and makes it possible to have many small and large data objects placed in the same simulated contiguous data space.

Once the data object has been placed in the simulated contiguous data space, it can be mapped to the correct disk pages based on information contained in the database directory.
25 However mapping an entire data object at its first reference is expensive and would require considerable real storage. For this reason this invention provides a mechanism to delay the mapping until first reference of a data object, and then to map only a segment consisting of the desired pages of the data object by a procedure called "mapping on demand."

Mapping on demand can be explained as follows: For each sub-data space a bit map is
30 created to indicate if each segment in the sub-data space is currently "referenced". A segment is referenced if the pages containing data in that segment have been mapped to disk. Initially, the bit map is set to '0' and the entire sub-data space is considered not referenced. The bit is referred

CA9-91-004B

10

to as the "segment valid bit" and one bit map is used per data space. For example if a sub-data space is made up of 524 288 pages and each segment is 256 pages, then there are 2048 segments and the bit map is therefore $2048/8 = 256$ bytes per sub-data space which is allocated when the sub-data space is created.

5 Although the entire data object is placed in the simulated contiguous data space when it is first referenced, the sub-data spaces and the bit maps may not all exist. The mechanism provided by this invention to calculate the page address of any page in the contiguous area detects if this is the first reference to a sub-data space and/or a segment of the data object. If this is the first reference, a sub-data space is created and a segment is mapped. The steps comprising mapping
10 on demand can be summarized as follows:

 Step 1. At first reference to a data object, place it in the simulated contiguous data space by allocating the starting and ending data space address as previously described.

 Step 2. Calculate the address of the referenced page as previously described.

 Step 3. If the sub-data space was not created, then create the sub-data space as previously
15 described, set the data space identifier value used to address that sub-data space, and allocate and initialize the segment validity bit map.

 Step 4. If the segment containing the page is not mapped, then map it and turn on the bit to indicate that the segment was mapped.

 This procedure is summarized in the following pseudo code:

20

If Starting_page_number for data object(n) is not assigned then

do;

 Starting_page_number(n) = Next_available_page;

 Reserved_space = ((Data_object_size(n)/(z) + 1) * (z);

25

 Next_available_page = Next_available_page + Reserved_

space;

end;

do;

 Data_space_number = (Starting_page_number(n) + m)/x;

30

 Page-address = ((Starting_page_number(n) + m)//x) * y;

end;

If Data_space_pointer(Data_space_number) was not created then

CA9-91-004B

11

do;

Create the data space

Set Data_space_pointer(Data_space_number) to the value
used to address that data space.

5 Allocate the segment valid bit map for that data space and initialize it to off. end;

If Segment_valid(Page_address / y*z) is off then

do;

Map the segment containing Page-m

10 Set segment_valid(Page_address / y*z) on;

end;

15 In the above pseudo code; Starting_page_number is an array of as many elements as the number of data objects in the database; (/) represent the division function; (//) represent the remainder of a division; (*) represent the multiplication operator; (z) is the number of pages in a segment; (y) is the size of a page in bytes; (m) is the referenced page from the start of the data object; (n) is the identity of the data object.

20 Checking the "segment valid" bit at each page request is inexpensive as the bit map is small. If the segment is not mapped, then the database manager program has to read the directory blocks containing information on the pages required for that segment and issue a request to map the data space pages that contain data to their locations on disk. Once mapped, the "segment valid bit" is turned on and any subsequent reference to read a page in that segment will be translated directly to a virtual storage address without reference to the database directory. The net result is an improved query response time.

25 If a request is made for an update (the page was modified), the page may have to be shadowed (moved to a new disk location); alternatively it may have to be inserted if it had not existed before (first insert). This will require a change to the mapping in which case the database manager will determine from the database directory if the page needs to be (re)mapped. If so, a new data page is allocated, the directory is updated and a new mapping is done. This will result
30 in a mapping change, not a change in the virtual address of a page.

An example of mapping on demand as provided by one aspect of the present invention can be illustrated using the IBM® SQL/DS™ database management system operating on an IBM

CA9-91-004B

12

System/390™ computer with the IBM VM/ESA™ operating system as follows. Before updating a page, the IBM SQL/DS management system will copy the requested page to a buffer. It will then re-issue the MAPDISK DEFINE request with the PAGEVIEW = ZERO parameter before copying the data back from the local buffer to the data space. Note, PAGEVIEW = ZERO informs VM/ESA operating system that it should provide an empty real storage page for that data space page rather than reading the contents of the disk when the page is first referenced.

If updating without the use of a local buffer is desired, the page can be remapped with the PAGEVIEW = RETAIN parameter. This first reads the copy of the page from its current disk (old) location before re-setting the mapping to the new location. Consequently the user is presented with the old copy of the page which will be written to the new location.

When the local buffer is used, the re-mapping can be deferred to when the page is moved back from the local buffer to the data space. Performance optimization is implemented by delaying the remapping in an attempt to group multiple mapping request into one MAPMDISK DEFINE call.

For this purpose, several MAPMDISK DEFINE request blocks can be initialized and used as required. Each request can be batched for several consecutive data space pages.

When a page is moved to the local buffer and a MAPMDISK DEFINE is required, the following is done for a new request:

i) If a MAPMDISK DEFINE request exists for the previous or the next page of the same data space, the two requests are combined into one.

ii) If the new request cannot be attached to any existing request so that it would be consecutive, and there exists an unused request block, use it to store the new request.

iii) If all the pre-allocated request areas are in use, issue the DEFINE for the requests already in one of the request control blocks and reset the block for the current page.

Since, by using an embodiment of the present invention, the operating system is now responsible for performing the database I/O operations through its paging mechanism, an interface is used to ensure that all modified data is written to the database at regular intervals, maintaining integrity of the database. Without such mechanism, integrity of the database.

In the prior art, mechanisms exist to request that changes to a database be written to disk while the database manager is waiting. However such mechanisms may cause significant delays at checkpoint time as many pages may be needed to be written to disk when data spaces are used. To resolve this problem, some operating system have implemented a mechanism whereby the

CA9-91-004B

13

database manager can specify a subset of the database to be made current on disk (i.e. to be written to disk if the pages in memory were modified). Further the mechanism is said to be asynchronous in that a save request can be issued, and processing continued, while the I/O operations are performed. An indicator is presented to the database manager when all I/O needed for that save have completed.

One embodiment of the present invention takes advantage of the above mechanism to implement a partial save. Like mapping on demand which allows the database manager to delay the mapping to the first reference of a page, the invention provides for a partial save which allows the database manager to request page saves at short intervals and for only the modified pages of a data object.

Partial save can be illustrated using the IBM SQL/DS database management system as follows:

Like the Segment Valid Bit map, a bit map is allocated and set to '0' when a data space is referenced for the first time. In this case the bit map is made of 1 bit for every 32 pages (or 4 096 bytes per data space) which we refer to as 'MODIFIED BIT MAP'. The bit representing a set of 32 pages can be calculated by dividing the page address by 131,072 (32 pages of 4 096 bytes each). Before moving a page from the local buffer back to the data space, the corresponding bit in the 'MODIFIED BIT MAP' is checked. If it is OFF (0), it is then turned ON (1) and a counter is incremented to keep track of the number of bits currently ON.

When the counter reaches a predefined limit (set by the user) the data base manager would issue a MAPMDISK SAVE using as the list of addresses to save, the list corresponding to the pages represented by the ON in the 'MODIFIED BIT MAP'. The database manager would then reset the bit map to OFF (0). Since the request is asynchronous, the database manager would continue processing while the I/O operations take place. The value of the counter determines the frequency with which the partial saves are issued (a lower counter causes more frequent partial saves). The counter also determines the maximum number of outstanding page sets for which the SAVE has been issued to save all the database pages on disk.

The steps comprising partial save can be summarized as follows:

Step 1. When a data space is created, allocate a bit map made of 1 bit per 'n' pages of data spaces and initialize the bit map to '0'. Initialize a counter (Modified_group_count) for that data space to 0.

Step 2. When modifying the page in the data space, check the corresponding bit in the bit

CA9-91-004B

14

map. If '0', increment the Modified_group_count for that data space and change the bit to '1'.

Step 3. If a Modified_group_count reaches the save interval threshold build a save request made of all the pages corresponding to the '1' bits in the bit map for that data space. Reset the bit map to '0' and issue the SAVE request as an asynchronous request.

5 Step 4. At checkpoint time, build a save request for each data space of the pages corresponding to the '1' bits in the bit map of the data space. Reset the bit map to '0' and issue the save request. Wait for all the request to complete before marking the checkpoint as completed.

The data space creation procedure of the present invention can be represented in pseudo code as shown as follows:

10

If Data_space_pointer(Data_space_number) = 0 (this data
space does not already exist) then

do;

Create the data space

15

Set Data_space_pointer(Data_space_number) to the ALET
value used to address that data space.

Allocate the segment valid bit map for that data
space and initialize it to off.

20

Allocate the section_is_modified bit map for that
data data space of size(m/n) and
initialize it to off.

Set Modified_group_count(Data_space_number) to 0.

end;

25

Similarly the page modification procedure can be represented in pseudo code as shown as follows:

Modify the page.

If Modified_group_count(Data_space_number) >
Save_interval

30

and Section_is_modified(page-address/(n*y) is off then
do;

Initialize a save request

CA9-91-004B

15

Do $i = 1$ to (m/n)

If `section_is_modified(i)` is on then

Add pages from $(i*n)$ to $((i*n)+n)$ to the save
request

5 end;

Reset `Section_is_modified` to off

Issue the save request without wait for completion

end;

10 If `Section_is_modified(page-address/(n*y))` is off then do; /* none of the pages in this
section were modified */ Increment `modified_group_count(Data_space_number)` by

1. Set `Section_is_modified(page-address/(n*y))` on.

End;

15 Similarly the checkpoint procedure can be represented in pseudo code as shown as
follows:

Do $j = 1$ to $\max(\text{Data\_space\_number})$

Initialize a save request

Do $i = 1$ to (m/n)

20 If `section_is_modified(i)` for data space(j) is on

Add pages from $(i*n)$ to $((i*n)+n)$ to the save
request

End;

Reset `Section_is_modified` for data space(j) to off

25 Issue the save request without wait for completion

End;

End;

Wait for all outstanding save requests to complete.

30 In the above pseudo code, (*) is the multiplication operator, (x) is the number of pages in
one data space; (y) is the number of bytes per page; (m) is the size in pages of the data object; (n)
is the number of pages represented by 1 bit; (i) is a counter.

CA9-91-004B

16

For each SAVE request issued, the data base manager would keep the control block built to issue the request until the request completes. On completion, if the operating system indicates that some of the I/O could not be done due to a hardware problem, the data base manager can take corrective action.

5 No restrictions are placed on pages that are referenced in a MAPMDISK SAVE request so that they can be modified after the request is issued even before the SAVE completion is received. No assumptions is made on what version of the page was written out, but the system will save the latest copy at a later time.

10 With this invention, the database manager I/O function is bypassed. Furthermore, page I/O is implemented so that the operating system informs the database manager that a paging I/O is needed (and will inform it when it is completed) allowing the database manager to perform other tasks in parallel with the I/O. This technique has been proven to result in very significant improvement in data base response time without compromising data integrity.

CA9-91-004B

17

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1 1. A method for modifying a page of a data object contained in a database and stored in one
2 or more database storage disks, wherein the database comprises one or more database data
3 objects, wherein the database is accessed via a contiguous data space representation, the
4 contiguous data space being represented by a plurality of concatenated sub-data spaces, wherein
5 each sub-data space is a virtual data space of a maximum size that is addressable by a computer
6 operating system, each sub-data space comprising a plurality of data segments, each data segment
7 comprising a plurality of pages, wherein the pages in the contiguous data space representation are
8 contiguous, each page comprising a known number of addressable storage locations, wherein the
9 contiguous data space, the sub-data spaces, the data segments, and the pages are addressable by a
10 database management system, wherein the address of a page to be modified has been determined
11 to be placed in a sub-data space, the method comprising the steps of:

12 (1) mapping a data segment of the data object from a database storage disk to a data
13 segment of the sub-data space, wherein said data segment from said database storage disk
14 contains the page to be modified;

15 (2) creating a modified section bit map for the sub-data space, wherein said modified
16 section bit map comprises a plurality of bits, each bit representing a group of pages of the
17 sub-data space and indicating whether one or more pages in said group has been modified;

18 (3) indicating that the page to be modified has been modified, thereby creating a modified
19 page, by setting a bit in said modified section bit map to a predetermined value, wherein said bit
20 represents a group containing said modified page;

21 (4) generating a modified section count representing a number of bits in said modified
22 section bit map that are equal to said predetermined value; and

23 (5) issuing an asynchronous request to save one or more groups of pages of the sub-data
24 space corresponding to bits in said modified section bit map that are equal to said predetermined
25 value when said modified section count corresponds to a modified section threshold.

1 2. A method for modifying a page of a data object contained in a database and stored in one
2 or more database storage disks, wherein the database comprises one or more database data
3 objects, wherein the database is accessed via a contiguous data space representation, the

CA9-91-004B

18

contiguous data space being represented by a plurality of concatenated sub-data spaces, wherein each sub-data space is a virtual data space of a maximum size that is addressable by a computer operating system, each sub-data space comprising a plurality of data segments, each data segment comprising a plurality of pages, wherein the pages in the contiguous data space representation are contiguous, each page comprising a known number of addressable storage locations, wherein the contiguous data space, the sub-data spaces, the data segments, and the pages are addressable by a database management system, wherein the address of a page to be modified has been determined to be placed in a sub-data space, the method comprising the steps of:

(1) mapping a data segment of the data object from a database storage disk to a data segment of the sub-data space, wherein said data segment from said database storage disk contains the page to be modified;

(2) creating a modified section bit map for the sub-data space, wherein said modified section bit map comprises a plurality of bits, each bit representing a group of pages of the sub-data space and indicating whether one or more pages in said group has been modified;

(3) indicating that the page to be modified has been modified, thereby creating a modified page, by setting a bit in said modified section bit map to a predetermined value, wherein said bit represents a group containing said modified page;

(4) generating a modified section count representing a number of bits in said modified section bit map that are equal to said predetermined value; and

(5) issuing an asynchronous request to save one or more groups of pages of the sub-data space corresponding to bits in said modified section bit map that are equal to said predetermined value when said modified section count corresponds to a modified section threshold;

wherein, if the page to be modified requires a new mapping to said database storage disk, the method comprising the further steps of:

(6) copying the page to be modified to another work space in computer memory and modifying the page to be modified in said another work space in computer memory;

(7) allocating a new database storage page in said database storage disk;

(8) determining whether a mapping request to map the page to be modified to said new database storage page in said database storage disk can be combined with a prior mapping request, wherein said prior mapping request maps a second page in the sub-data space to a second database storage page in said database storage disk and the page to be modified is adjacent to said second page in the sub-data space;

CA9-91-004B

19

(9) issuing said prior mapping request if step (8) determines that said mapping request cannot be combined with said prior mapping request and said prior mapping request exists;

(10) copying said second page of said prior mapping request from said another work space in computer memory if step (9) issues said prior mapping request;

(11) preparing said mapping request if step (9) issues said prior mapping request or said prior mapping request does not exist; and

(12) combining said mapping request with said prior mapping request if step (8) determines that said mapping request can be combined with said prior mapping request.

3. The method of claim 1, further comprising the step of:

(6) issuing a synchronous request to save one or more groups of pages of each said sub-data space corresponding to bits in each said modified section bit map that are equal to said predetermined value when the database management system determines that a checkpoint is required, wherein the database management system waits for all outstanding asynchronous requests to complete.

4. A computer system for modifying a page of a data object contained in a database and stored in one or more database storage disks, wherein the database comprises one or more database data objects, wherein the database is accessed via a contiguous data space representation, the contiguous data space being represented by a plurality of concatenated sub-data spaces, wherein each sub-data space is a virtual data space of a maximum size that is addressable by a computer operating system, each sub-data space comprising a plurality of data segments, each data segment comprising a plurality of pages, wherein the pages in the contiguous data space representation are contiguous, each page comprising a known number of addressable storage locations, wherein the contiguous data space, the sub-data spaces, the data segments, and the pages are addressable by a database management system, wherein the address of a page to be modified has been determined to be placed in a sub-data space, comprising:

mapping means for mapping a data segment of the database data object from a database storage disk to a data segment of the sub-data space, wherein said data segment from said database storage disk contains the page to be modified;

bit map means for creating a modified section bit map for the sub-data space, wherein said modified section bit map comprises a plurality of bits, each bit representing a group of pages

CA9-91-004B

20

of the sub-data space and indicating whether one or more pages in said group has been modified;

indicating means for indicating that the page to be modified has been modified, thereby creating a modified page, by setting a bit in said modified section bit map to a predetermined value, wherein said bit represents a group containing said modified page;

generating means for generating a modified section count representing the number of bits in said modified section bit map that are equal to said predetermined value; and

issuing means for issuing an asynchronous request to save one or more groups of pages of the sub-data space corresponding to bits in said modified section bit map that are equal to said predetermined value when said modified section count corresponds to a modified section threshold.

5. The computer system according to claim 4, wherein if the page to be modified requires a new mapping to said database storage disk, further comprising:

first copying means for copying the page to be modified to another work space in computer memory and modifying the page to be modified in said another work space in computer memory;

allocating means for allocating a new database storage page in said database storage disk;

determining means for determining whether a mapping request to map the page to be modified to said new database storage page in said database storage disk can be combined with a prior mapping request, wherein said prior mapping request maps a second page in the sub-data space to a second database storage page in said database storage disk and the page to be modified is adjacent to said second page in the sub-data space;

issuing means for issuing said prior mapping request if said determining means determines that said mapping request cannot be combined with said prior mapping request and said prior mapping request exists;

second copying means for copying said second page of said prior mapping request from said another work space in computer memory if said issuing means issues said prior mapping request;

preparing means for preparing said mapping request if said issuing means issues said prior mapping request or said prior mapping request does not exist; and

combining means for combining said mapping request with said prior mapping request if said determining means determines that said mapping request can be combined with said prior

CA9-91-004B

21

22 mapping request.

1 6. The computer system according to claim 4, further comprising:

2 second issuing means for issuing a synchronous request to save one or more groups of
3 pages of each said sub-data space corresponding to bits in each said modified section bit map that
4 are equal to said predetermined value when the database management system determines that a
5 checkpoint is required, wherein the database management system waits for all outstanding
6 asynchronous requests to complete.

1 7. A program storage device readable by a machine tangibly embodying a computer
2 program product for use with a database management system, comprising:

3 a computer usable medium having computer readable program code means embodied in
4 said medium for modifying a page of a data object contained in a database and stored in one or
5 more database storage disks, wherein the database comprises one or more database data objects,
6 wherein the database is accessed via a contiguous data space representation, the contiguous data
7 space being represented by a plurality of concatenated sub-data spaces, wherein each sub-data
8 space is a virtual data space of a maximum size that is addressable by a computer operating
9 system, each sub-data space comprising a plurality of data segments, each data segment
10 comprising a plurality of pages, wherein the pages in the contiguous data space representation are
11 contiguous, each page comprising a known number of addressable storage locations, wherein the
12 contiguous data space, the sub-data spaces, the data segments, and the pages are addressable by a
13 database management system, wherein the address of a page to be modified has been determined
14 to be placed in a sub-data space, said computer program product having:

15 first computer readable program code means for mapping a data segment of the database
16 data object from a database storage disk to a data segment of the sub-data space, wherein said
17 data segment from said database storage disk contains the page to be modified;

18 second computer readable program code means for creating a modified section bit map
19 for the sub-data space, wherein said modified section bit map comprises a plurality of bits, each
20 bit representing a group of pages of the sub-data space and indicating whether one or more pages
21 in said group has been modified;

22 third computer readable program code means for indicating that the page has been
23 modified, thereby creating a modified page, by setting a bit in said modified section bit map to a

CA9-91-004B

22

predetermined value, wherein said bit represents a group containing said modified page;

fourth computer readable program code means for generating a modified section count representing the number of bits in said modified section bit map that are equal to said predetermined value; and

fifth computer readable program code means for issuing an asynchronous request to save one or more groups of pages of the sub-data space corresponding to bits in said modified section bit map that are equal to said predetermined value when said modified section count corresponds to a modified section threshold.

8. A program storage device readable by a machine tangibly embodying a computer program product for use with a database management system, comprising:

a computer usable medium having computer readable program code means embodied in said medium for modifying a page of a data object contained in a database and stored in one or more database storage disks, wherein the database comprises one or more database data objects, wherein the database is accessed via a contiguous data space representation, the contiguous data space being represented by a plurality of concatenated sub-data spaces, wherein each sub-data space is a virtual data space of a maximum size that is addressable by a computer operating system, each sub-data space comprising a plurality of data segments, each data segment comprising a plurality of pages, wherein the pages in the contiguous data space representation are contiguous, each page comprising a known number of addressable storage locations, wherein the contiguous data space, the sub-data spaces, the data segments, and the pages are addressable by a database management system, wherein the address of a page to be modified has been determined to be placed in a sub-data space, said computer program product having:

first computer readable program code means for mapping a data segment of the database data object from a database storage disk to a data segment of the sub-data space, wherein said data segment from said database storage disk contains the page to be modified;

second computer readable program code means for creating a modified section bit map for the sub-data space, wherein said modified section bit map comprises a plurality of bits, each bit representing a group of pages of the sub-data space and indicating whether one or more pages in said group has been modified;

third computer readable program code means for indicating that the page has been modified, thereby creating a modified page, by setting a bit in said modified section bit map to a

CA9-91-004B

23

predetermined value, wherein said bit represents a group containing said modified page;

fourth computer readable program code means for generating a modified section count representing the number of bits in said modified section bit map that are equal to said predetermined value; and

fifth computer readable program code means for issuing an asynchronous request to save one or more groups of pages of the sub-data space corresponding to bits in said modified section bit map that are equal to said predetermined value when said modified section count corresponds to a modified section threshold;

wherein, if the page to be modified requires a new mapping to said database storage disk, said computer program product further having:

sixth computer readable program code means for copying the page to be modified to another work space in computer memory and modifying the page to be modified in said another work space in computer memory;

seventh computer readable program code means for allocating a new database storage page in said database storage disk;

eighth computer readable program code means for determining whether a mapping request to map the page to be modified to said new database storage page in said database storage disk can be combined with a prior mapping request, wherein said prior mapping request maps a second page in the sub-data space to a second database storage page in said database storage disk and the page to be modified is adjacent to said second page in the sub-data space;

ninth computer readable program code means for issuing said prior mapping request if said eighth computer readable program code means determines that said mapping request cannot be combined with said prior mapping request and said prior mapping request exists;

tenth computer readable program code means for copying said second page of said prior mapping request from said another work space in computer memory if said ninth computer readable program code means issues said prior mapping request;

eleventh computer readable program code means for preparing said mapping request if said ninth computer readable program code means issues said prior mapping request or said prior mapping request does not exist; and

twelfth computer readable program code means for combining said mapping request with said prior mapping request if said eighth computer readable program code means determines that said mapping request can be combined with said prior mapping request.

CA9-91-004B

24

1 9. A program storage device readable by a machine tangibly embodying the computer
2 program product according to claim 7, further comprising:

3 sixth computer readable program code means for issuing a synchronous request to save
4 one or more groups of pages of each said sub-data space corresponding to bits in said modified
5 section bit map that are equal to said predetermined value when the database management system
6 determines that a checkpoint is required, wherein the database management system waits for all
7 outstanding asynchronous requests to complete.

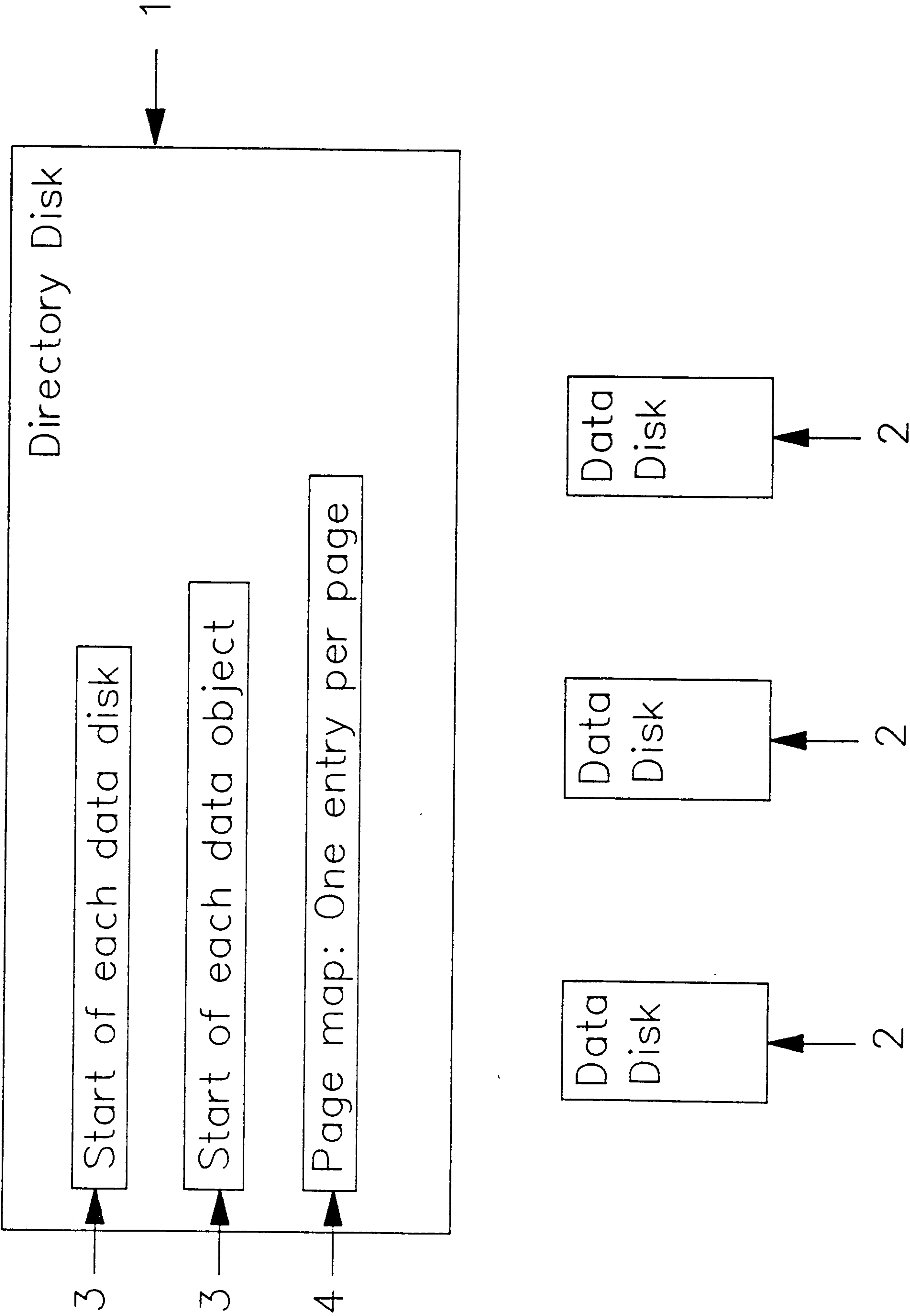


Fig 1

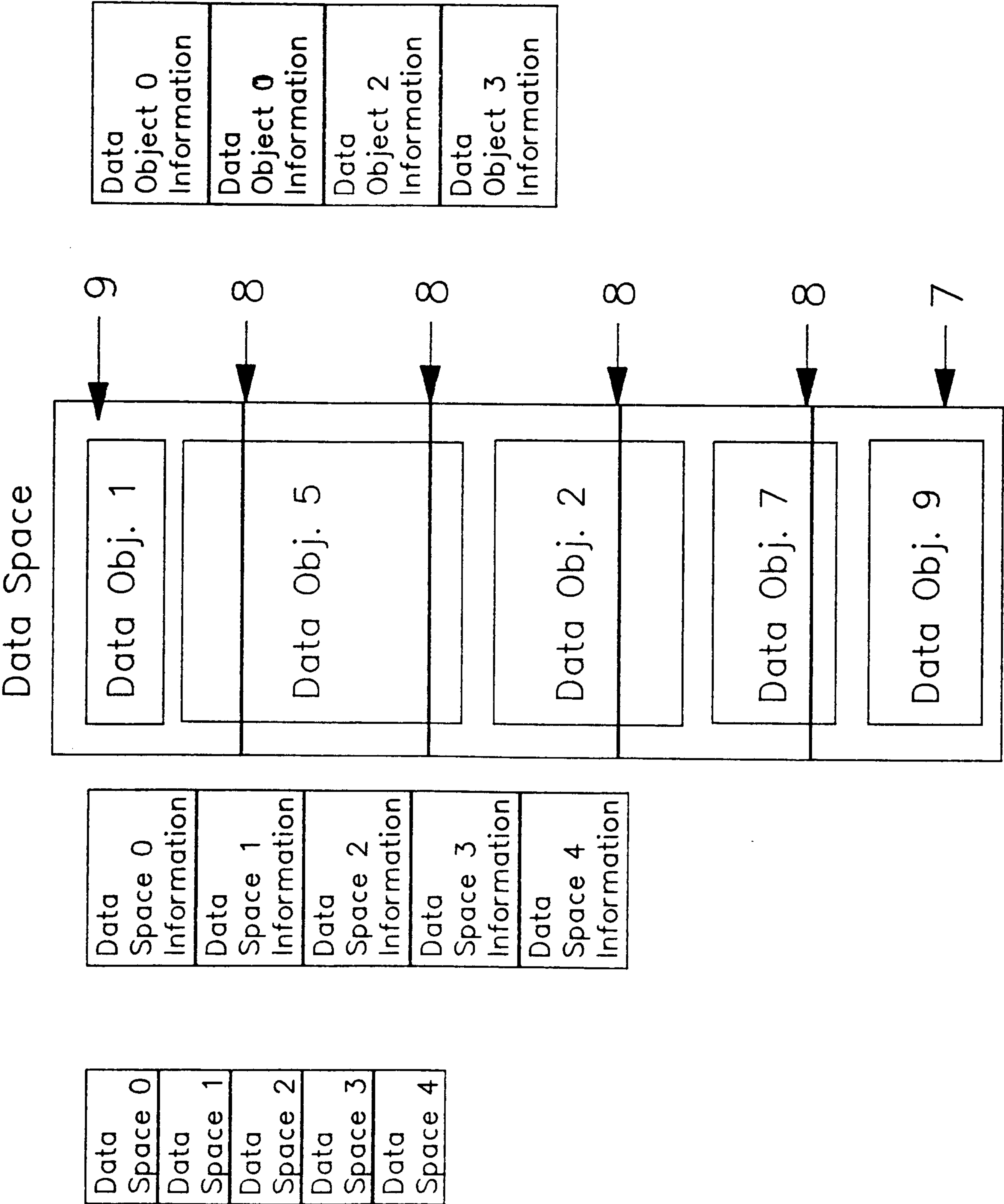


Fig. 2

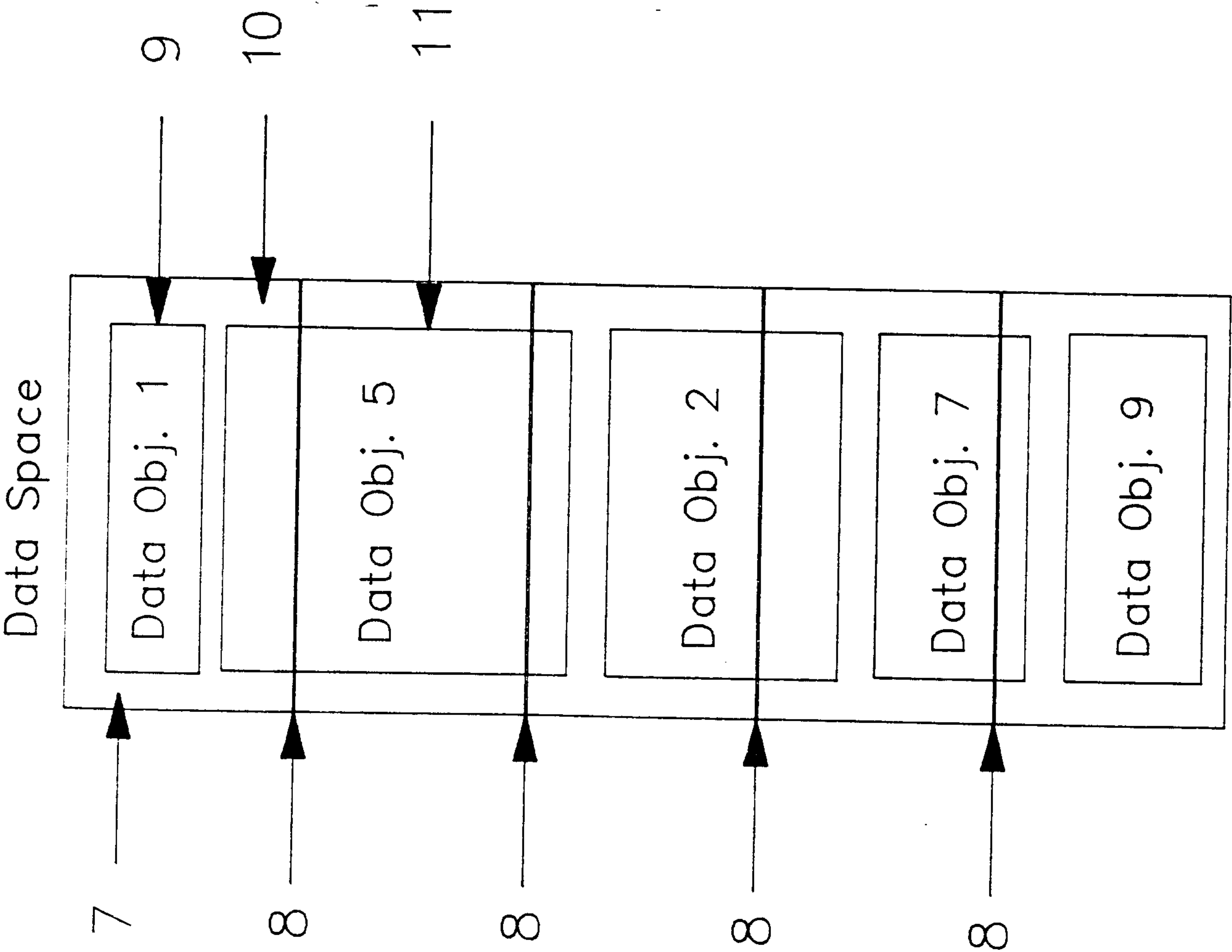


Fig. 3

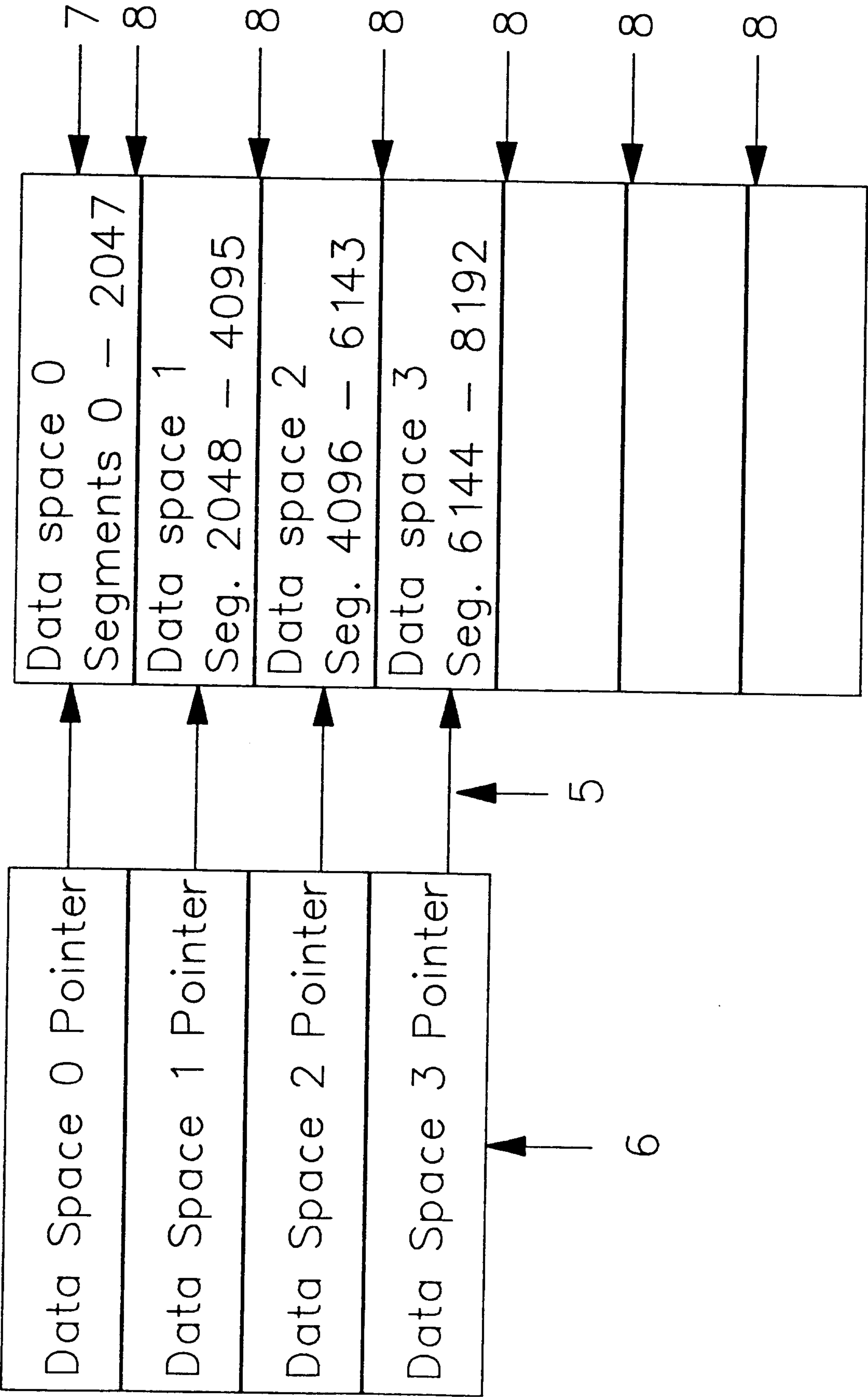


Fig 4

Data Space No.		Page No. In Data Space			
Segment Number					
0000	0000	0000	0000	0000	0000

Fig 5