UCID- 18912

MASTER

SYSTEM DATA STRUCTURES FOR ON-LINE DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

J. A. WADE

JANUARY 28, 1981

Lawrence Livermore Laboratory

CONTENTS

SYSTEM DATA STRUCTURES FOR ON-LINE DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

SUMMARY

Described herein are the data structures used in implementing a distributed data base management system (DBMS) for the Mirror Fusion Test Facility (MFTF), a part of the Mirror Fusion Energy Program at the Lawrence Livermore National Laboratory. The hardware and software frameworks within which the DBMS have been developed are first described, followed by a brief look at the motivation and fundamental design goals of the system. The structures are then given in detail.

INTRODUCTION

A DBMS that has been specifically adapted for MFTF--rather than specialized data manipulation routines incapable of future expansion--has been designed and in the process of implementation. Simply defined, it is a comprehensive set of software tools and documentation which facilitate access to the MFTF Data Base. Programs, operating on Interdata computers, provide capabilities for creation of the Data Base, monitoring its usage, offloading the Data Base to archival storage and subsequent reloading from archival storage, recovery when catastrophic events occur, and access to the Data Base from interactive terminals. Libraries of subroutines, along with other support packages, provide the user with access to the Data Base from computer programs performing specific functions which require MFTF data as input and possibly creating additional data to be written back to the Data Base. Documentation in the form of descriptive material, instructions for use of the various facets of the DMBS, and illustrative examples provide knowledge to use the DBMS. Generally, the program and library parts of the DBMS may be decomposed into four main aspects:

a. Pre-compiler - Used to transform Data Base and other constructs existing in source code.

b. Program Level Interface (PLI) - Consists of a run-time library of procedures and functions, and associated support programs.

c. Query Level Interface (QLI) - Used for access to the Data Base by a user from an interactive terminal.

d. Utility Programs - Perform a variety of support functions.

HARDWARE FRAMEWORK

Pictured in Figure 1, the DBMS is implemented on set of nine Interdata computers (four 8/32s and five 7/32s) that are interconnected in a star fashion with a multi-port shared memory.[1] Each computer has its own local memory and disk storage; the 7/32s each have a 10-megabyte disk, and the 8/32s each have an 80-megabyte disk with one 300-megabyte disk and an additional 10-megabyte disk installed on one of the 8/32s. Two of the 8/32s each have a 1600-bpi, 75-ips tape drive. Shared memory is arranged on two 64-kilobyte blocks. Seven of the nine computers incorporate MFTF operator consoles, designed to perform specific MFTF functions. Two of the computers, the System Supervisor and the Injector Supervisor, can each cause any of the other computers to bootstrap the operating system (OS) from disk; hence remote OS restart can be done. Although not shown in this figure, several bus switches exist so that should (really, when) one computer goes down, another can take over its functions by switching the satellite subsystems.

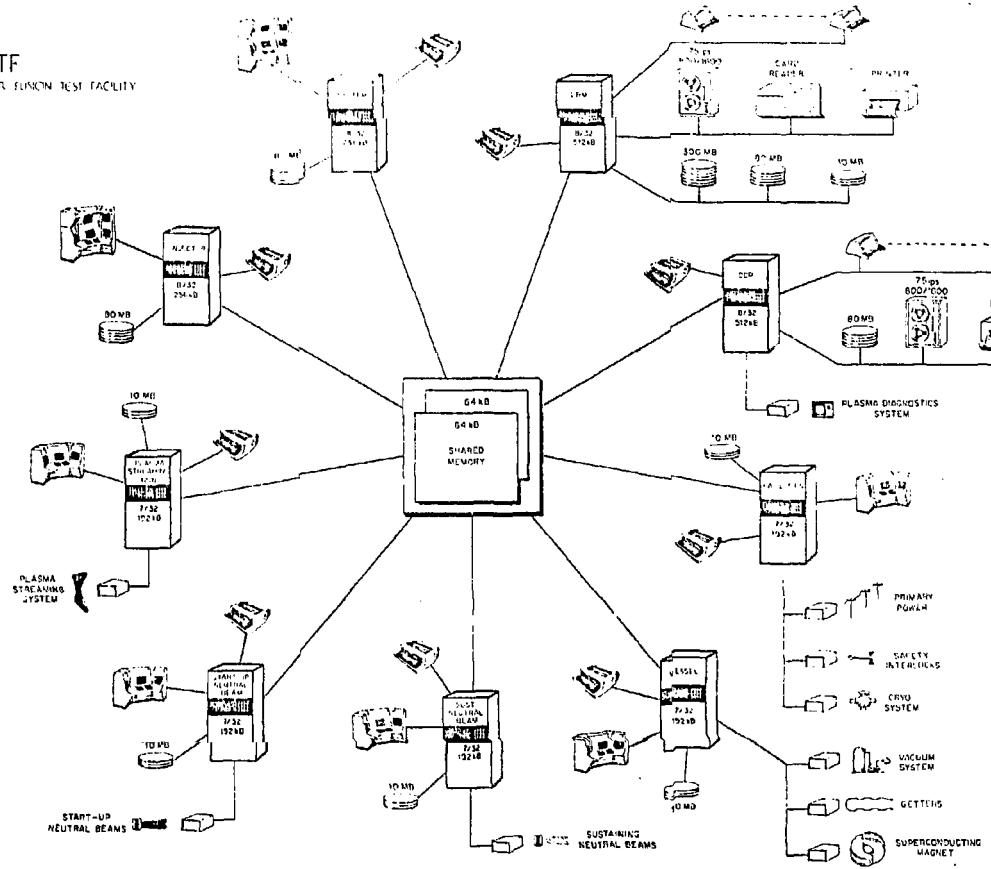


FIGURE 1: MFTF CONTROL AND DIAGNOSTICS SYSTEM

SOFTWARE FRAMEWORK

There are several issues which work together to form the software framework for the DBMS. First of all the Interdata-supplied operating system was extended to a multi-processor OS in a manner so as to provide availability of resources, common storage management, process control, mutual exclusion, and associated error handling.[2] Secondly, the shared memory is logically divided in two, with the distributed OS managing one half and the distributed DBMS managing the other half. Concerning the OS portion of shared memory, buffers are allocated and deallocated dynamically and are used to contain mail, of which there are three types: semaphores, commands, and data. Semaphores are used by the OS for its own synchronization and by the DBMS to maintain synchronization among its multi-user concurrent community. Commands are used by the DBMS for inter-process requests, whether the process be on the same physical machine or not. Concerning the portion of shared memory managed by the DBMS, it is divided up into two basic entities; summary description space, and page space. Further aspects of the actual data structures resident in shared memory are presented later.

Another issue which effects the DBMS is our use of the implementation language Pascal.[3] Acquired from Kansas State University, it is the sequential version of Brinch Hansen's Concurrent Pascal,[4] along with a few "enhancements" to allow further capabilities.

Finally, several programs have been built to aid in development of the software packages. Most notably is an editor which builds finished versions of documentation according to IBM's HIPO techniques,[5] printed on a Versatec printer. Additionally, systematics for code generation, source-level documentation standards, and structured walkthru techniques have been used during construction of the software.

FUNDAMENTAL DESIGN ISSUES

Early in the preliminary design iteration of the overall MFTF control system project, a decision was made that all data of importance to MFTF is to be contained in the data Base. This declared that both control-type data and diagnostics-type data be resident in the Data Base, each of which display different properties concerning speed of access and volume of data. The control data encompasses a large volume of scalar data (single data values) such as current temperatures, pressures, valve positions and journaling information requiring much higher speed access than is normally avialable in a commercial DBMS. In contrast, the diagnostics data is composed of a lessor volume (with respect to count of relations), but much larger vectors (arrays of data values). Due to the large volume of data acquired from each MFTF shot (four megabytes of diagnostics alone for each five minute shot cycle), the DBMS must be able to respond efficiently to both types of volume requests. In addition, the scalar-oriented control data generally requires higher-speed access than the vector-oriented diagnostic data. Obviously, a number of mechanisms were necessary to allow the Data Base Administrator to "tune" the Data Base, based on specific speed versus volume tradeoffs.

SYSTEM DATA STRUCTURES

The DBMS run-time library (RTL) performs several functions. At the user level, routines exist to initialize a program to the DBMS (a one-time operation), create, delete, open, close, read, write, checkpoint, lock, unlock, and search tables. Although initialization, opening and closing of tables are allowed to utilize a fairly "large" amount of processor time, Data Base reads and writes must consume as little time as is reasonable. The DBMS data structures have been specifically designed to that end.

RTL I/O HIERARCHY

Figure 2 shows the overall hierarchy of RTL I/O organization. Given, for example, a user-level read request from the Data Base, the table's state variably contained in the user's impure stack space, is passed to the RTL which uses it to access various system tables, eventually copying the specific data from the DBMS work space area into the user's memory space. The state variable is also updated as a result.

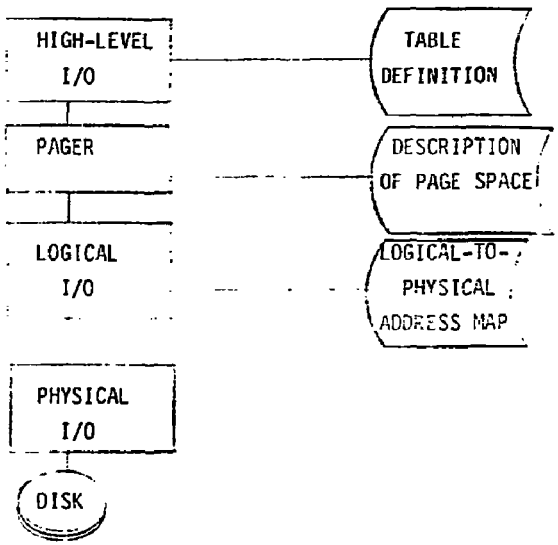


Figure 2. Hierarchical View of RTL I/O Routines.

DATA BASE ACCESS PATHS

Referencing Figure 3, a user task never needs to access an external task to perform reads or writes on that part of the Data Base resident on the same physical machine. Additionally, if the data is in DBMS page space, no I/O is requested.

Should the data requested for read reside on a physical machine other than the machine upon which the user's program is running, the RTL read routine forms an appropriate command and passes it to a "DBMS task" on the machine upon which the data resides. This surrogate actually reads the Data Base and returns the data, along with the updated table state variable.

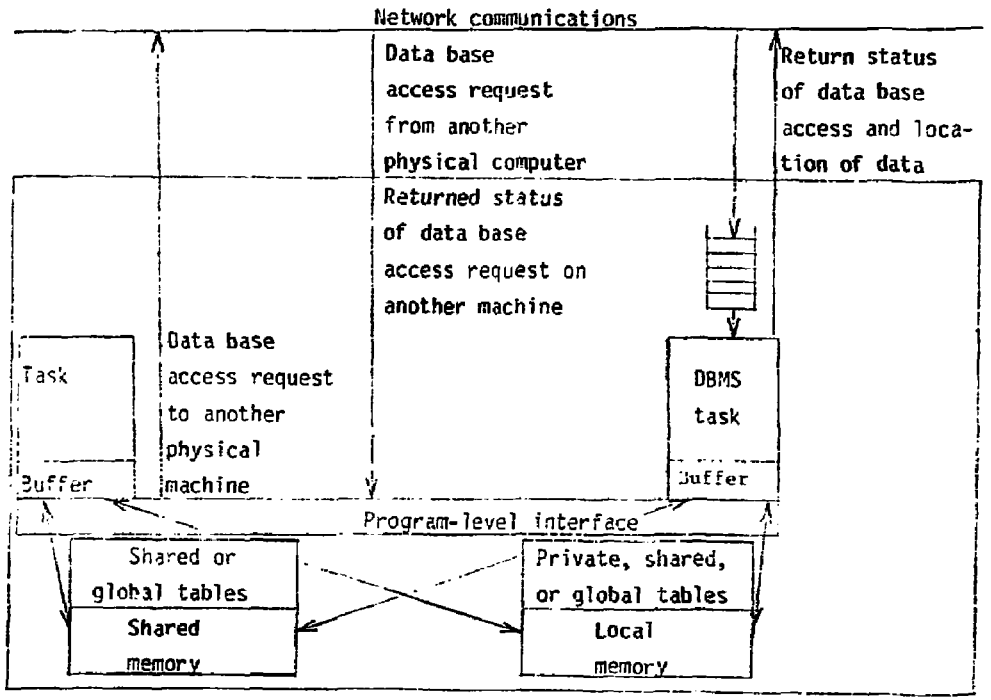


Figure 3. Data Base Access Paths for a Physical Computer.

Although the above is not meant to provide a complete description of what functions the RTL performs and how they are actually done, enough of a flavor is provided to form the framework for discussing the System Data Structures in detail.

SHARED MEMORY DATA BASE SPACE

Referring to Table 1A, the SMDB space contains several sizes, pointers, locks and other data used throughout the distributed DBMS for its overall coordination. In addition to a basic size of SMDB Space, its associated lock, and total available page count, there are four head-of-chain entries used for various concepts. At this time, only two of the items are used; further development of the system will complete use of the other links. Beyond the four chain entries lie the set of shared memory page descriptions and related page data space. Described in detail elsewhere, these pages fully support concurrent intra- and inter-machine functions.

Common Size:

Specified originally as an input parameter to the initial DBMS startup procedure, this value is checked for validity and entered in memory as the size (in bytes) of the total available Shared Memory Data Base space.

Common Lock:

Allocated during the startup process, this semaphore is never used by the DBMS. Its ultimate intention is as a hook for quiescing the entire DBMS.

Page Count:

The count of SMDB pages available to the DBMS. Set as a function of common size.

Save Relation List Lock:

Save Relation List Lock LA:

Although the semaphore is intially allocated, this lock and its associated Logical Address are never actually used in the DBMS. Space is reserved for future expansion. The intention is to provide a central head-of-chain entry to those relations that must be saved on archive medium at some future time.

Open Relation List Lock:
Open Relation List Lock PN:

This head-of-chain entry, along with its semaphore, provide access to the list of all relations currently open (ie., currently being accessed) in the DBMS. Page Number refers to a page in SMDB Space which is the first block of open relation entries.

Delete Relation List Lock:
Delete Relation List Lock LA:

Although the semaphore is initally allocated, this lock and its associated Logical Address are never acutally used in the DBMS. Space is reserved for future expansion. The intention is to provide a central Head-of-chain entry to those relations that have been marked as deleted; actual reclamation of the space is suggested to be done during "idle" time.

SMDB Pages Lock:

This semaphore controls access to the set of SMDB Page Descriptions existing elsewhere in SMDB. When a page is desired, its Logical Address is looked for in the page descriptions only after this overall semaphore has been activated.

Top of LRU Page Queue:
Bottom of LRU Page Queue:

Contain the corresponding page number of the top and bottom of the LRU (Last Recently Used) page queue. The most recent page used is at the top; the last recent page used is at the bottom.

RTE Free Space Lock:
RTE Free Space PH:

Controls access and points to the head-of-chain for available Run Time Entry Free Space. Since RTEs consume half-pages, the pointer is a physical page and half-page number pair of the first available RTE Free Space Entry. These RTEs are dynamically allocated and released from SMDB only and are used to contain information about the more dynamic state of particular relations currently known to the system. Upon release, joined half-pages are returned to the SMDB page space.

Page Descriptions:

Page Data Space:

Consisting of one entry per defined and known SMDB page, the page descriptions contain information specific to the particular corresponding data page, one description entry per data space page.

TABLE 1A. Shared Memory Data Base (SMDB) Space.

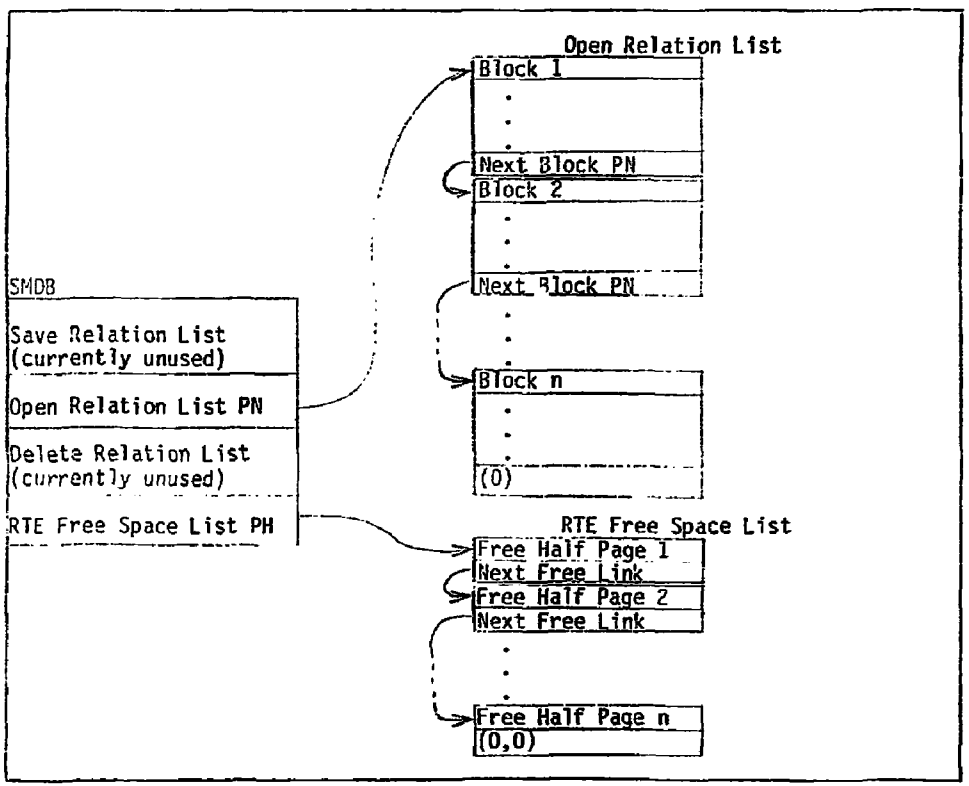
Common Size
Common Lock
Page Count
Save Relation List Lock
Save Relation List LA
Open Relation List Lock
Open Relation List PN
Delete Relation List Lock
Delete Relation List LA
SMDB Pages Lock
Top of LRU Page Queue
Bottom of LRU Page Queue
RTE Free Space Lock
RTE Free Space PN

Page Descriptions	Page Data Space
Page 1 Description	Page 1 Data Space
Page 2 Description	Page 2 Data Space
.	.
.	.
.	.
Page n Description	Page n Data Space

Link Summary

As an alternate view of SMDB Data Space, Table 18 shows the link structure of the two defined head-of-chain descriptions, the Open Relation List and RTE Free Space List. As stated previously, both Save and Delete Relation Lists are currently unused, but with the head-of-chains defined. As this link structure shows, both defined lists have their entries blocked into "sets" to be nearly approximate to the size of a page or half-page data space; a link to the next set of entries is the last element in a block. The link is a physical page number, a logical address, or a page, half-page combination depending on whether the block is swappable to disk or not and the probable number of entries per block (so as to minimize empty space).

TABLE 18. Shared Memory Data Base (SMDB) Space



Open Relations List

Table 1C shows the individual description for a typical block of entries in the Open Relation List. Referenced by either the head-of-chain Open Relation List value in SMDB or a Next Block page number, the blocks together comprise all relations being accessed in the distributed DBMS at a specific point in time. Each block may or may not be full; empty entries are intermixed with entries currently in use. The reasoning is that compaction of used entries is too expensive for such a dynamic process. Each time a relation is opened for use, this chain is searched to see if it is already open by another user. If already open, the rather expensive open when not yet used need not be done. In this system, it is the binding of relation name with a specific owner which provides uniqueness to the relation; that is to say that there may be several relations called, R, but each with a different owner. Upon an open for a relation, the Run Time Entry address is copied to the user's state variable for that relation for subsequent use and the open count is incremented, Careful comparison of this data structure with others will show that specific user may open the same relation numerous times; when an interaction is terminated, all this is cleared up.

Relation Name:
Name of the relation which is currently open.

Relation Owner:
Name of owner of the currently open relation.

RTE Location:
SMDB page number and half-page where the RTE is located.

Open Count:
Total count of users currently accessing this relation.

TABLE 1C. Open Relation List

SMDB - Open Relation List - Block i				
	Relation Name	Relation Owner	RTE Location	Open Count
Entry 1				
Entry 2				
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Entry n				
Next Block PN				

Page Descriptions

Given that SMDB contains a set of pages that may contain various data, there is one description for each page in SMDB, as shown in Table 1D; the i-th page description contains information regarding the i-th corresponding data page. At any point in time, certain of the data pages (and therefore the associated page description(s)) may or may not be included in the LRU Queue. For example, the Open Relation List is a set of linked SMDB pages not in the LRU Queue, but the head of which is in SMDB. The RTE Free Space List and the actual RTEs themselves are additional examples of such behavior. Although the total page count in SMDB is explicitly specified, the actual current page count (which is the total *minus* those used for the Open Relation List, etc.) is implicit in the LRU Queue.

Page Info Lock:

This semaphore is locked when the corresponding data page is currently in use. During the same time, other pages may also be locked.

IO Lock:

This semaphore is locked via a PC (P-concurrent) when read access is desired, and via a PS (P-sequential) when a write access is desired for the corresponding data page.

Logical Address:

If the corresponding data page contains valid data for some portion of logical data space (either for a relation's data or for symbol table data), this contains the logical address of the page. If less than or equal to zero, the page is not within logical space (in which case, the page is not in the LRU Queue either).

Owned Machine:

Existing only for SMDB pages, this differentiates among several pages in SMDB belonging to various machines. A page is owned by a specific machine if and only if the Owned Machine is the same as the LMDB machine identifier.

Update Type:

Set to "yes" if and only if the corresponding page of data has been modified since its last write to disk. Changed to "no" during a successful page flush to disk.

Page Type:

Set to "virtual" if the corresponding data page is a member of a relation's data space and may be reused for other data space should the need require. Set to "contained" if the page may not be reused for other data space. Set to "workspace" if the page is being used by the DBMS for its data space. If "virtual", the page description exists in the LRU Queue; otherwise it does not.

Previous Page:

Page number of the immediately newer page in the LRU Queue. Equals -1 if the current page is at the top of the LRU Queue.

Next Page:

Page number of the immediately older page in the LRU Queue. Equals -1 if the current page is at the bottom of the LRU Queue.

TABLE 1D. Shared Memory Page Map

SMDB - Page Description								
	Page Info Lock	IO Lock	Logical Address	Owned Machine	Update Type	Page Type	Previous Page	Next Page
Page 1								
Page 2								
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
Page n								

LOCAL MEMORY DATA BASE SPACE

In contrast to SMDB Space (for which only one copy exists in Shared Memory), LMDB Space is defined on each physical computer in the distributed system. Although the structure of LMDB, shown in Table 2A, is the same throughout the DBMS, the data resident therein is unique per physical machine. As with SMDB, there is a common size, lock, total available page count, and page descriptions with related page data space. In addition, several scalar values, pointers and associated locks reside in LMDB describing the current state of the DBMS of the particular machine.

Common Size:

Specified as an input argument to the DBMS task which initialized the DBMS for a given machine, this value is checked for validity and entered in memory as the size (in bytes) of the total available Local Memory Data Base Space.

Common Lock:

Allocated during the startup process, this semaphore is never used by the DBMS. Its ultimate intention is as a hook for quiescing the DBMS on the specific physical machine.

Page Count:

The count of LMDB pages available to the DBMS. Set as a function of common size.

Machine Identifier:

Physical machine designation of the computer upon which this part of the DBMS is running.

Active Task List Lock:

Active Task List LA:

Controlled with its semaphore, the Active Task List is a complete list of all tasks (programs) currently accessing the Data Base. The Logical Address (as it is named) is actually the LMDB page number of the head of the chain.

Logical Space Free Space Lock:

Logical Space Free Space LA:

Controlled with its semaphore, this Free Space Logical Address is the logical location of the free space map for relation data space. As further data space is allocated or deallocated for containment of data for a relation, the map referenced by this logical address is updated.

Symbol Table Free Space Lock:

Symbol Table Free Space LA:

Controlled with its semaphore, this Free Space Logical Address is the logical location of the free space map for DBMS symbol table space. Noting that relation data space and DBMS symbol table space both map onto the same set of physical files the symbol table space is allocated and deallocated as a function of relations being created and deleted.

Last Physical File Number Lock:

Lasy Physical File Number LA:

As further physical file space is needed, contiguous files of a specific length are allocated. The new data space is added to the free-space pool of relation data or symbol table map and this last physical file number is incremented.

Last Logical Space Address Used Lock:

Last Logical Space Address Used:

As further relation data space is allocated or deallocated, this semaphore-controlled value is updated.

Last Symbol Table Address Used Lock:

Last Symbol Table Address Used:

As further symbol table space is allocated or deallocated, this semaphore-controlled value is updated.

Hash Table Lock:

Controls access to the hash table, resident in LMDB page number one. Accessed, among elsewhere, upon relation open requests when a relation is found to be not yet open anywhere.

LMDB Pages Lock:

This semaphore controls access to the set of LMDB Page Descriptions existing elsewhere in Local Memory. When a page is desired, its Logical Address is looked for in the page descriptions only after this overall semaphore has been activated.

Top of LRU Page Queue:

Bottom of LRU Page Queue:

Contain the corresponding page number of the top and bottom of the LRU (Last Recently Used) page queue. The most recent page used is at the top; the last recent page used is at the bottom.

Page Descriptions:

Page Data Space:

Consisting of one entry per defined and known LMDB page, the page descriptions contain information specific to the particular corresponding data page, one description entry per data space page.

TABLE 2A. Local Memory Data Base (LMDB) Space

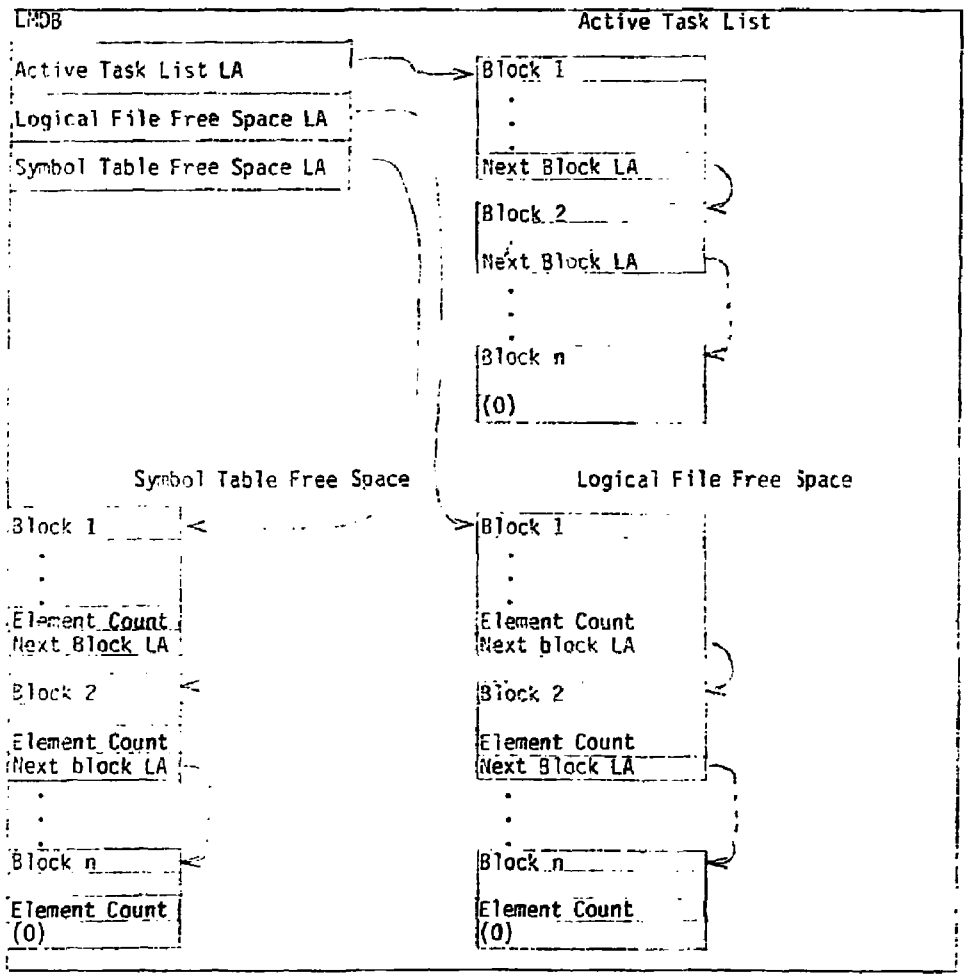
Common Size
Common Lock
Page Count
Machine Identifier
Active Task List Lock
Active Task List PN
Logical Space Free Space Lock
Logical Space Free Space LA
Symbol Table Free Space Lock
Symbol Table Free Space LA
Last Physical File Number Lock
Last Physical File Number
Last Logical Space Address Used Lock
Last Logical Space Address Used
Last Symbol Table Address Used Lock
Last Symbol Table Address Used
Hash Table Lock
LMDB Pages Lock
Top of LRU Page Queue
Bottom of LRU Page Queue

Page Descriptions	Page Data Space
Page 1 Description	Page 1 Data Space
Page 2 Description	Page 2 Data Space
⋮	⋮
Page n Description	Page n Data Space

Link Summary

Again, in comparision to Table 1B, Table 2B shows the link structure of LMDB Data Space. Whereas the links shown in 1B are completely memory contained, only the Active Task List in 1C (LMDB) is memory contained; the two free space lists shown reside on disk - only their initial pointers are in memory. Similiar to other lists elsewhere in the DBMS, the entries are blocked into "sets" with each set (in this case) consuming one page of LMDB page space.

TABLE 2B. Local Memory Data Base (LMDB) Space



Active Task List

Table 2C shows the individual description for a typical block of Active Task List entries; its head-of-chain physical LMDB page number is contained in Table 2A's overall contents with access controlled via semaphore. There may exist empty entries (denoted by a blank task name) in one or more of the blocks. The last block's next block page number is zero. The entire set of entries (ie., all entries in the list) are quite dynamic; as tasks start or terminate interaction with the Data Base, one or more entries in this list are modified.

Task Name:

Name of the task (program) accessing the Data Base.

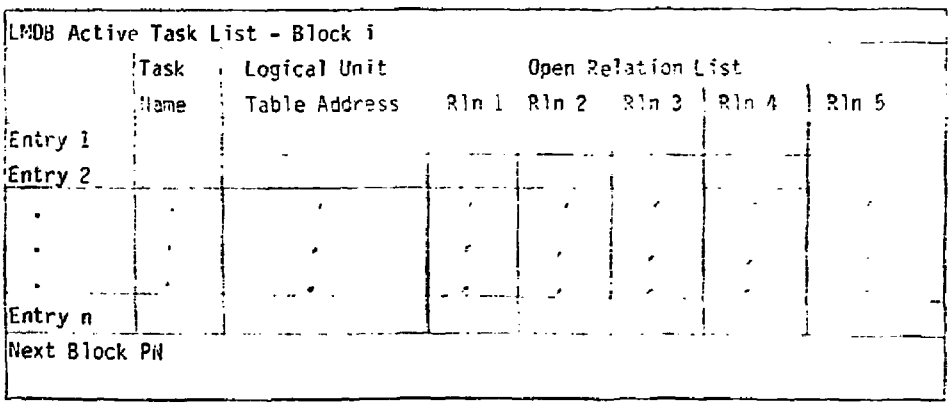
Logical Unit Table Address:

Memory Address of the DBMS Logical Unit Table used to issue disk I/O directly from the user task. The DBMS uses units 16-31.

Open Relation List:

A set of up to five page number, half-page combinations pointing to an active Run Time Entry (in SMDB), one per active relation. Should more than five relations be concurrently open by the user in one session, a second entire Active Task List entry is generated.

TABLE 2C. Active Task List



Page Descriptions

Given that LMDB contains a set of pages that may contain various data, there is one description for each page in LMDB; the i-th page description contains information regrding the i-th corresponding data page. At any point in time, certain of the data pages (and therefore the associated page description(s)) may or may not be included in the LRU Queue. For example, the Active Task List is a set of linked LMDB pages not in the LRU Queue, but the head of which is in LMDB. The Hash table, resident as LMDB page one and statistics gathering space as LMDB pages two and three are additional examples of such behavior. Although the total page count in LMDB is explicitly specified, the actual current page count (which is the total minus those used for the Active Task List, etc.) is implicit in the LRU Queue.

Page Info Lock:

This semaphore is locked when the corresponding data page is currently in use. During the same time, other pages may also be locked.

IO Lock:

This semaphore is locked via a PC (P-concurrent) when read access is desired, and via a PS (P-sequential) when a write access is desired for the corresponding data page.

Logical Address:

If the corresponding data page contains valid data for some portion of logical data space (either for a relation's data or for symbol table data), this contains the logical address of the page. If less than or equal to zero, the page is not within logical space (in which case, the page is not in the LRU Queue either).

Update Type:

Set to "yes" if and only if the corresponding page of data has been modified since its last write to disk. Changed to "no" during a successful page flush to disk.

Page Type:

Set to "virtual" if the corresponding data page is a member of a relation's data space and may be reused for other data space should the need require. Set to "contained" if the page may not be reused for other data space. Set to "workspace" if the page is being used by the DBMS for its data space. If "virtual", the page description exists in the LRU Queue; otherwise it does not.

Previous Page:

Page number of the immediately newer page in the LRU Queue. Equals -1 if the current page is at the top of the LRU Queue.

Next Page:

Page number of the immediately older page in the LRU Queue. Equals -1 if the current page is at the bottom of the LRU Queue.

TABLE 20. Local Memory Page Map

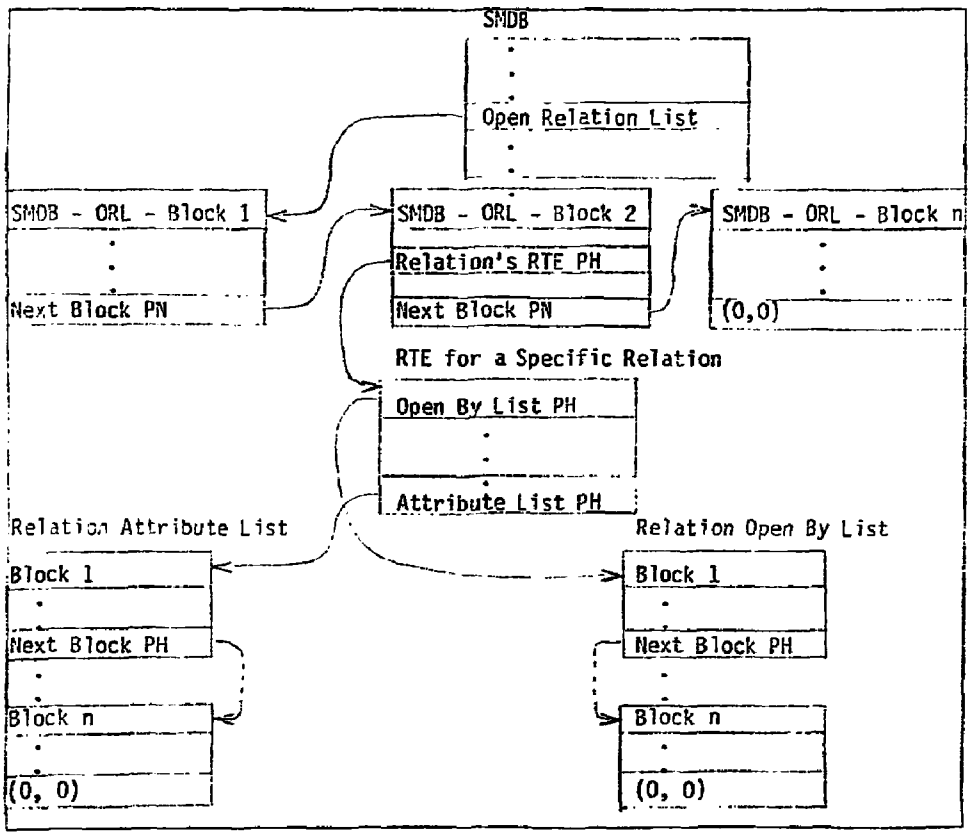
LMDB - Page Descriptions

	Page Info Lock	Logical Address	Update Type	Page Type	Previous Page	Next Page	IO Lock
Page 1							
Page 2							
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
Page n							

RUN TIME ENTRIES

As referenced briefly in the discussions for Tables 1A, 1B and 2C, there are several places in DBMS Data Structures pointing to Run Time Entries (RTEs). Given a task name for a specific physical machine, one can reference the Active Task List and obtain the RTE addresses of all relations being currently referenced for that task. Alternatively, the Open Relation List (in SMDB) may be referenced to obtain a comprehensive list of all relations currently active in the system. RTEs are the fundamental unit of description for a relation. From its root, two other lists are available (specific to the particular relation), the relation's Open By List and the relation's Attribute List. All three data structures are memory contained in SMDB and consume half pages only.

TABLE 3A. Run Time Entry for a Relation (Link Summary)



Main Entry for a Relation

Referring to Table 3B, Run Time Entries are maintained in SMDB and are memory contained (ie., not pageable), each consuming one-half page each. To find an RTE for a specific (relation, owner) pair, the Open Relation List is searched for satisfaction of criterion. The RTE pointer is then used to access the particular SMDB page and half-page. When a user requests to open a relation, if the relation is not yet open, the RTE is built in SMDB space using STEs for the relation and its associated additional pages of data. Once built, the RTE is maintained until all access to the relation is completed, at which time the used half-pages are returned to the RTE Free Space List.

Open User List:

Page, half-page combination used to list all users currently accessing the particular relation.

Owner Private Access:

Set of rights the owner possesses for restricting his own access to the relation.

Global Access:

Set of rights given to all other users who do not own the relation nor are given explicit access to the relation via the share list concept.

Primary Frame Location:

Starting Logical Address of the primary copy of the primary copy of the relation.

Primary Frame Size:

Count of bytes necessary to contain the primary copy of the relation.

Primary Frame Machine:

Physical machine upon which the disk files containing the primary copy of the relation reside. Should I/O be required for access to the relation, it must be done on this machine.

Duplicate Frame Machine:

Physical machine upon which the disk files containing the duplicate copy of the relation reside. Should I/O be required for access to the relation, it must be done on this machine.

Duplicate Frame Logical Address:

Starting Logical Address of the duplicate copy of the relation.

Duplicate Frame Size:

Count of bytes necessary to contain the duplicate copy of the relation.

Memory Location Type:

States whether the relations is utilizing local (LMDB) or shared (SMDB) page space.

Memory Usage Type:

States whether the relation is memory-contained or virtual. If memory-contained, all data for the relation is paged in with the pages set as not eligible for reuse.

Read Count:

Incremented by one for each read request against the relation.

Write Count:

Incremented by one for each write request against the relation.

Open Count:

Incremented by one for each open request against the relation. When this count equals zero, there are no tasks currently accessing the particular relation.

Tuple Size:

Count of bytes required to contain one full tuple.

Tuple Access Method:

Data structure method required for access to a tuple.

Tuple Access Size:

Used to aid the Tuple Access Method.

Lower Tuple id:

Lowest tuple id the relation may reach.

Bottom Tuple id:

Currently defined lowest tuple id.

Top Tuple id:

Currently defined highest tuple id.

Upper Tuple id:

Highest tuple id the relation may reach.

Ring Size:

If the Tuple Access Method is of a ring type, this is the maximum number of tuples that can be defined at one time.

Notify List:

Not documented herein.

Attribute Access Method:

Data Structure method required for access to attributes within a tuple.

Attribute List PH:

Page number and half of the start of all defined attributes for the relation.

Attribute Count:

Count of defined attributes.

TABLE 3B. Run Time Entry for a Relation

SMDB - Run Time Entry
Open User List
Owner Private Access
Global Access
Primary Frame Location
Primary Frame Size
Primary Frame Machine
Duplicate Frame Machine
Duplicate Frame Logical Address
Duplicate Frame Size
Memory Location Type
Memory Usage Type
Read Count
Write Count
Open Count
Tuple Size
Tuple Access Method
Tuple Access Size
Lower Tuple id
Bottom Tuple
Top Tuple
Upper Tuple
Ring Size
Notify List
Attribute Access Method
Attribute List PH
Attribute Count

Open By List

Table 3C shows the particular data definition for a block of entries for a relation. Pointed intially to by an RTE element, the entries comprise all currently defined users of the relation in question. In order to maintain high speed access to currently opened relation, the access rights of all users for open relations are maintained in memory so that subsequent opens by shared users incur minimum overhead. The Next Block page number, half combination equals (0, 0) for the last defined block.

User Name:

Ascii name of a potential shared user.

User Access Rights:

Set of rights restricting access to the relation for the potential user.

TABLE 3C. RTE Relation Open By List

SMDB - Relation Open by List		
	User Name	User Access Rights
Entry 1		
Entry 2		
.	.	.
.	.	.
.	.	.
Entry n		
Next Block Page, Half		

Attribute List

Given that a relation contains attributes, Table 3D shows the detail for a given block of attribute descriptions. Initially pointed to from the relation's RTE in SMDB with a page number, half-page combination, the attribute entries are maintained in preorder form. Due to the nature of the pre-compiler versus run-time library concept, only the starting and ending byte address need be used during the run-time portion of data base access.

Start Byte Address:

Starting byte address of the corresponding attribute, relative to the start of a tuple.

End Byte Address:

Last defined byte address for the corresponding attribute, relative to the start of a tuple.

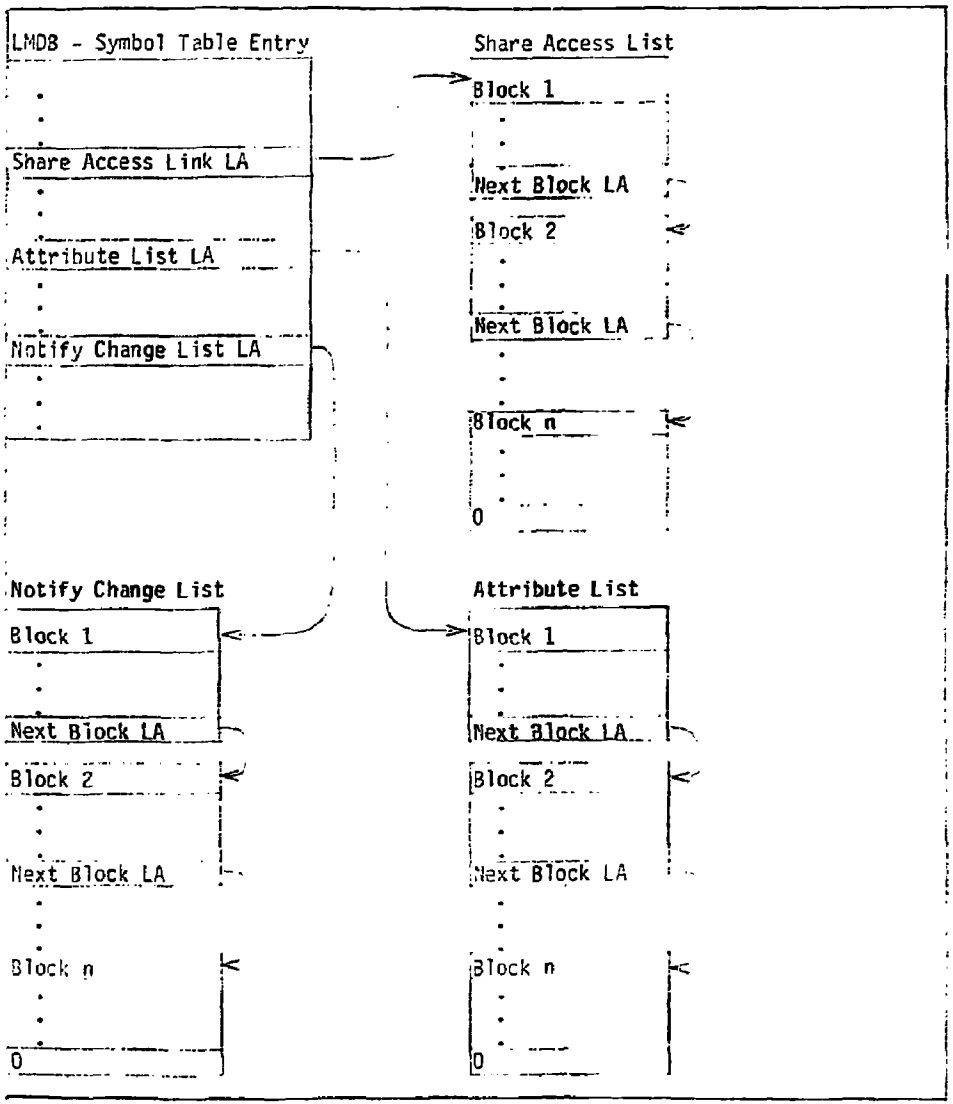
TABLE 3D. RTE Relation Attribute List

SMDB - Attribute List		
	Start Byte Address	End Byte Address
Entry 1		
Entry 2		
.	.	.
.	.	.
.	.	.
Entry n		
Next Block Page, Half		

SYMBOL TABLE ENTRIES

As relations are created, various Symbol Table Entries (STEs) are established to retain the information. Table 4A shows the link structure for a single realtion. Given a relation name, a hash value is generated and the hash table is probed (in LMDB, page 1) yielding either a null entry or a logical address for the head-of-collision chain list of all main STEs hashing to the same value. Branching out from the relation's main entry are the Attribute List, Share Access List, and Notify Change List. As with other lists throughout the DMBS, the entries are blocked into "sets" so that relatively full pages of data space may be used. All links from an STE to the defined lists are in terms of logical address; the actual data resides on disk (but may at some point in time be resident in LMDB page space).

TABLE 4A. Symbol Table Entry for a Relation
(Link Summary)



Main Entry for a Relation

Referring to Table 4B, it is this main Symbol Table Entry for a relation that is established during the create process. All other information is linked from this entry as its root. Given a relation name, a hash value is generated, the hash table (in LMDB, page 1) is probed, and a logical address is obtained. Via the paging mechanism, the associated page is read from disk and accessed. In this case, we are assuming that the RTE entries for the relation are being created in SMDB. Should the paged STE not be the main entry for the particular relation, the hash collision link is used as the logical address for an STE of another relation having the same hash value. Once the correct main STE is found for the relation, various logical address entries exist within it to link to the other STE types.

Relation Name:

Name of the relation.

Relation Owner:

Name of the relation's owner.

Hash Collision LA:

Used by the hashing mechanism to form a linked list of main STEs of all relations having the same hash value.

Owner Private Access:

Set of rights the owner possesses for restricting his own access to the relation.

Global Access:

Set of rights given to all other users who do not own the relation nor are given explicit access to the relation via the share list concept.

Share Access List LA:

Logical Address of the start of all users allowed access rights other than those allowed for the owner and globally.

Primary Location Information:

Structure containing information about the relation's primary copy of the data.

Create Date:

Structure denoting the create date of the relation.

Create Time:

Structure denoting the create time of the relation.

Frame Location:

Starting Logical Address of the relation.

Frame Size:

Count of bytes necessary to contain the relation.

Frame Prior Link:

Currently unused.

Frame Machine:

Physical machine upon which the disk files containing the relation reside. Should I/O be required for access to the relation, it must be done on this machine.

Memory Location Type:

States whether the relation is utilizing local (LMDB) or shared (SMDB) page space.

Memory Usage Type:

States whether the relation is memory-contained or virtual. If memory-contained, all data for the relation is paged in with the pages set as not eligible for reuse.

Tuple Access Size:

Used to aid the Tuple Access Method.

Tuple Access Method:

Data structure method required for access to attributes within a tuple.

Attribute Access Method:

Data structure method required for access to attributes within a tuple.

Duplicate Location Information

Structure containing information about the relation's duplicate copy of the data. Same as the Primary Location Information contents.

Lower Tuple id:

Lowest tuple id the relation may reach.

Bottom Tuple id:

Currently defined lowest tuple id.

Top Tuple id:

Currently defined highest tuple id.

Upper Tuple id:

Highest tuple id the relation may reach.

Ring Size:

If the Tuple Access Method is of a ring type, this is the maximum number of tuples that can be defined at one time.

Attribute List LA:

Logical Address of the start of the attribute descriptions for the relation.

Change Notify List LA:

Not documented herein.

Archive Link LA:

Currently unused.

STE Create Date, Time:

Date and Time when this main STE was created.

Prior Incarnation LA:

Currently unused.

Attribute Count:

Count of defined attributes.

Existence Status:

Current state of the relation; declared, created, or deleted.

TABLE 4B. Symbol Table Entry for a Relation

LYD3 - Symbol Table Entry
Relation Name
Relation Owner
Hash Collision Link LA
Owner's Private Access
Global Access
Share Access List LA
Primary Location Information

Create Date	Memory Location Type
Create Time	Memory Usage Type
Frame Location	Tuple Access Size
Frame Size	Tuple Access Method
Frame Prior Link	Attribute Access Method
Frame Machine	

Duplicate Location Information
Lower Tuple id
Bottom Tuple id
Top Tuple id
Upper Tuple id
Ring Size
Attribute List LA
Notify Change List LA
Archive Link LA (currently unused)
STE Create Date
STE Create Time
Prior Incarnation LA (currently unused)
Attribute Count
Existence Status

Attribute List

Table 4C shows the structure for a particular "set" or block of attributes for a relation, initially linked to from the relations main STE. Should more than one set be required, the Next Block Logical Address value points to it. As with the corresponding RTE data structure (Table 3D), the attributes are maintained in preorder form.

Attribute Name:

Name of the attribute.

Data Type:

Data type of the attribute.

Start Address:

Starting byte address of the attribute, relative to the start of a tuple.

Data Size:

Count of bytes necessary to contain the attribute's data.

Dimension Count:

Count of dimensions for the attribute, if the attribute is an array form. The attribute's data type then refers to the data type of an array element.

Dimensions:

Array (up to four elements allowed) of lower and upper bound combinations. Each combination corresponds to the i-th dimension.

TABLE 4C. STE Attribute List

LMDB - Attribute List							
	Attribute Name	Data Type	Start Address	Data Size	Dimension Count	Dimensions Lower Bnd	Upper Bnd
Entry 1							
Entry 2							
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
Entry n							
Next Block Logical Address							

Shared User Access List

Table 4D shows the data structure for a block of shared access users for a relation. Initially referenced by a logical address in the main STE of a relation. Next Block Logical Address points to the next block of shared users. As is the case with other STE types, zero is used to denote no further blocks.

Shared User:

Ascii name of a shared user.

Access Rights:

Set of rights restricting access to the relation for the shared user.

TABLE 4D. STE Share Access List

LMDB - Share Access List		
	Shared User	User Access Rights
Entry 1		
Entry 2		
.	.	.
.	.	.
.	.	.
Entry n		
Next Block Logical Address		

DBMS FILES

Throughout the previous discussion, logical addresses have been used. Specific to each physical machine is the concept of logical address space which is then mapped onto physical files. To introduce this idea, a description of what data exists in what files is presented. Further details as to specific structures follow.

Data Base space on disk is resident in several files, all of which use the file extension DBM.

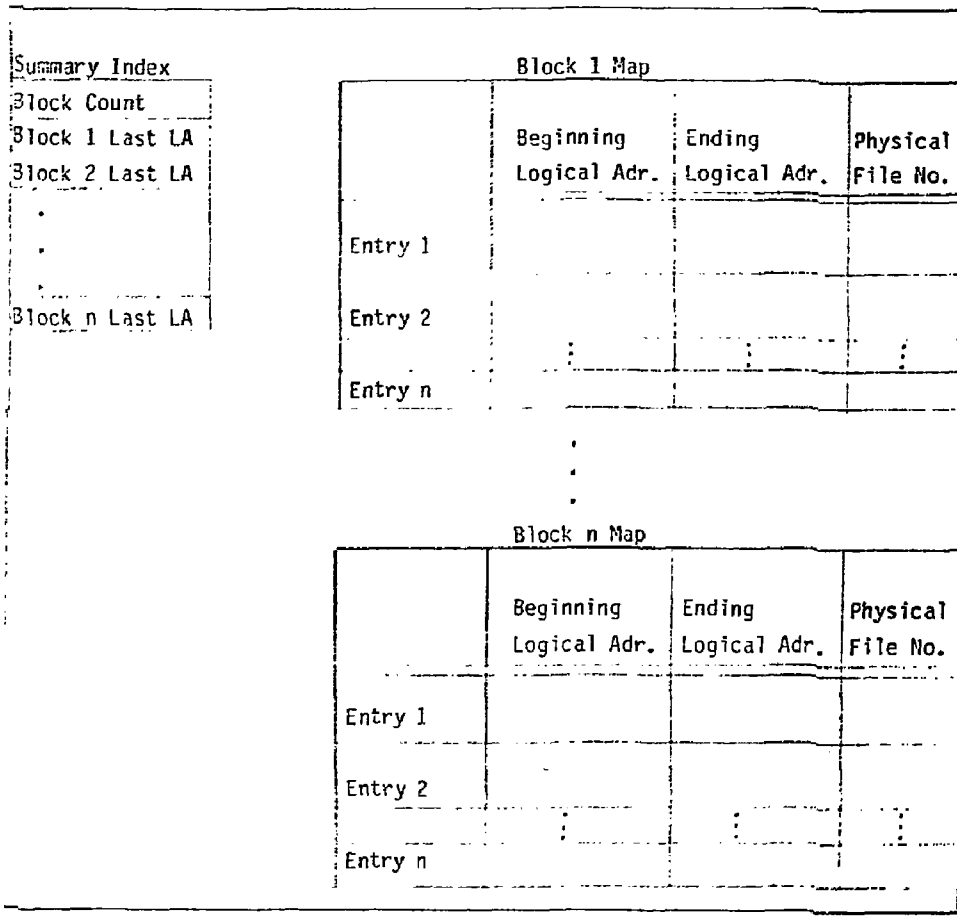
SMFILE.DBM

Currently written during every periodic page flush on all machines, this file is not yet fully developed. Its intention is to provide a file which contains information specific to SMDB, such as the save relation and delete relation lists.

PF000000.DBM

Referring to Table 5A, a summary index exists as the first sector in the file; each element in the index occupies four bytes. Following the index are the space transformation maps, one block per sector. Within a specific map, there is one entry, triplet per defined physical file containing a physical file number and its corresponding beginning and ending logical addresses; each four bytes long. To determine what physical file a logical address exists in, the summary index is first searched to determine which block to further search. The appropriate block is then accessed and a search is executed to determine the physical file number. The resultant file number is used to generate a file name (if necessary) of the form PFxxxxxx.DBM where xxxxxx is the file number.

TABLE 5A. Physical to Logical Address Transformation



PF000001.DBM

Referring to Table 5B, sectors zero through seven contain various data about the specific physical machine upon which the files reside. The numbers in parenthesis refer to how many bytes are used to contain the data. Many of the values defined herein exist in LMDB.

Physical Machine id:

Identifies the owning machine of the files.

Start LA of Relation Data Free Space Map:

Location of the space map of as yet unused data space for the symbol table.

Physical File Count:

Number of currently used physical files.

Last Relation Data Space LA Used:

Highest logical address of currently used relation data space.

Last Symbol Table Space LA Used:

Highest logical address of currently used symbol table space.

Hash Table:

Used for primary access to the symbol table, given a relation name.

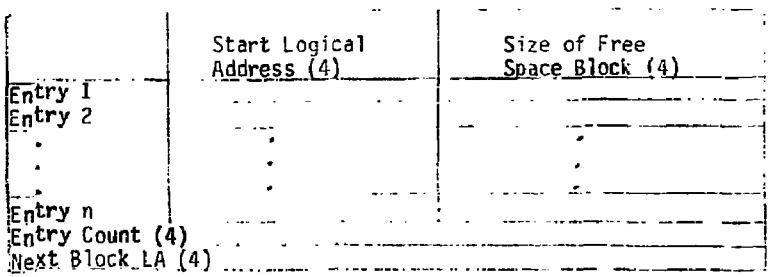
Statistics Data Space:

Currently unused. Intended for use in gathering frequency statistics regarding performance of the DBMS.

Relation Data Free Space Map:

Symbol Table Free Space Map:

Contain information regarding where free space currently exists for the entities. As with other linked lists, the last entry is the Next Block LA.



Since the pager works with logical addresses, these free space maps are subject to the same allowances given to all other data which is accessed primarily via logical address.

Start of Used Symbol Table Space:
The start of Logical Address space for the Symbol Table.

PF000001.DBM..PFnnnnnn.DBM

The physical data files for the DBMS on the specific physical machine. All logical address space maps into these disk files. The value nnnnnn is the Ascii representation of the Hex Integer value.

Referring to Figure 3, please note that the Data Base is accessed directly from a user task if the actual data exists on the physical machine upon which the task is running (assuming of course that I/O must be issued). This implies that the standard mechanism supplied by a vendor operating system to perform I/O from a user task via "logical units" be used; sufficient locks must exist to prevent concurrence problems. In addition, the DBMS logical to physical transformations just discussed must be taken into account.

TABLE 5B. Disk Space Utilization for PF000001.DBM

Sector Number	Description
0:	Physical Machine id (2)
	Unused (2)
	Start LA of Relation Data Free Space Map (4)
	Start LA of Symbol Table Free Space Map (4)
	Physical File Count (4)
	Last Relation Data Space LA Used (4)
	Last Symbol Table Space LA Used (4)
1:	Hash Table, 4 bytes entry, 61 entries
2,3:	Statistics Data Space
4,5:	Relation Data Free Space Maps
6,7:	Symbol Table Free Space Maps
8..:	Start of Used Symbol Table Space

DBMS LOGICAL UNITS FOR A USER

Table 6 shows the Logical Unit Table, local to a given user's task. Logical unit numbers 16-31 are reserved for DBMS usage. By way of example, say the page space for a relation is not in SMDB or LMDB (on the same physical machine) and must therefore be read in from disk. Once the logical address is determined, this table is searched, the physical disk address is calculated, and I/O issued against the affected logical unit.

TABLE 6. Logical Unit Table for a Task

	Start Logical Address	Physical File Size	Open Count
Entry 16			
Entry 17			
.	.	.	.
.	.	.	.
.	.	.	.
Entry 31			

STATE VARIABLE FOR A RELATION

Within a user's program, there are generally many places where Data Base references exist and the current "state" of access with respect to each relation must be retained. If, for example, a Data Base access exists in some isolated routine, the relation is "passed" to that routine (actually only the state variable is passed). Referring to Table 7, the state variable, per relation, contains several items.

Relation Error Status:

Contains the type and resultant error of Data Base access that was last done.

Relation Name:

Name of the Relation.

Owner Name:

Name of the owner

User Name:

Name of the user.

Current Tuple id:

Which tuple within the relation that was last accessed.

Current Attribute Number:

Which attribute within the tuple that was last accessed.

Function id:

Function last performed.

Line Number:

Pascal source line number where the Data Base access was called from.

Error Number:

Zero if no error; otherwise the number of the error for the function last performed.

Logical Unit Table Memory Address:

Memory location where the LU Table (see Table 6) exists.

Relation RTE Pointer:

Location of the main RTE for the relation (see Tables 3A-D).

User Access Rights:

Set of access restrictions applied to the specific user for the relation.

Workspace Tuple id:
Workspace Current LA:
Workspace Start:

Used by the DBMS for internal operations, dependant on the function performed.

TABLE 7. State Variable for a Relation

Relation Error Status	
Relation Name	Current Attribute Number
Owner Name	Function ID
User Name	Line Number
Current Tuple ID	Error Number
Logical Unit Table Memory Address	
Relation RTE Pointer (Page, Half)	
User Access Rights	
Workspace Tuple ID	
Workspace Current Logical Address	
Workspace Start Logical Address	

REFERENCES

1. Butner, D. N., "MFTF Supervisory Control and Diagnostics System Hardware", Proceedings of the Eighth IEEE Symposium on Engineering Problems of Fusion Research, (1979).

2. McGoldrick, P. R., "SCDS Distributed System," EPFR.

3. Young, Robert, PASCAL/32 Language Definition, Department of Computer Science, Kansas State University, (1978).

4. Brinch, Hanson P., "The Programming Language Concurrent Pascal," IEEE Transactions on Software Engineering, Vol. SE-1, No. 2, pp. 199-207 (June 1975).

5. HIPO-A Design Aid and Documentation Technique, 2nd Ed. (May 1975), GC20-1851-1, IBM.

Physical Machine id:

Identifies the owning machine of the files.

Start LA of Relation Data Free Space Map:

Location of the space map of as yet unused data space for the symbol table.

Physical File Count:

Number of currently used physical files.

Last Relation Data Space LA Used:

Highest logical address of currently used relation data space.

Last Symbol Table Space LA Used:

Highest logical address of currently used symbol table space.

Hash Table:

Used for primary access to the symbol table, given a relation name.

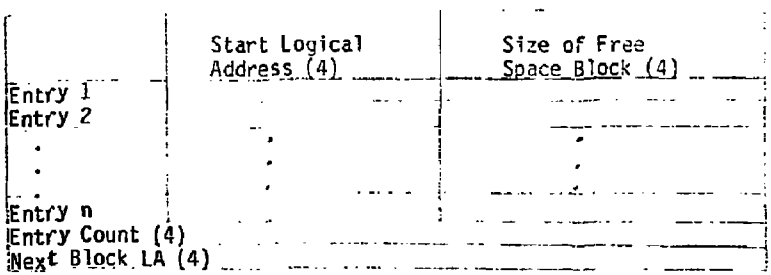
Statistics Data Space:

Currently unused. Intended for use in gathering frequency statistics regarding performance of the DBMS.

Relation Data Free Space Map:

Symbol Table Free Space Map:

Contain information regarding where free space currently exists for the entities. As with other linked lists, the last entry is the Next Block LA.



Since the pager works with logical addresses, these free space maps are subject to the same allowances given to all other data which is accessed primarily via logical address.

Start of Used Symbol Table Space:
The start of Logical Address space for the Symbol Table.

PF000001.DBM..PFnnnnnn.DBM

The physical data files for the DBMS on the specific physical machine. All logical address space maps into these disk files. The value nnnnnn is the Ascii representation of the Hex Integer value.

Referring to Figure 3, please note that the Data Base is accessed directly from a user task if the actual data exists on the physical machine upon which the task is running (assuming of course that I/O must be issued). This implies that the standard mechanism supplied by a vendor operating system to perform I/O from a user task via "logical units" be used; sufficient locks must exist to prevent concurrence problems. In addition, the DBMS logical to physical transformations just discussed must be taken into account.

TABLE 5B. Disk Space Utilization for PF000001.DBM

Sector Number	Description
0:	Physical Machine id (2)
	Unused (2)
	Start LA of Relation Data Free Space Map (4)
	Start LA of Symbol Table Free Space Map (4)
	Physical File Count (4)
	Last Relation Data Space LA Used (4)
	Last Symbol Table Space LA Used (4)
1:	Hash Table, 4 bytes entry, 61 entries
2,3:	Statistics Data Space
4,5:	Relation Data Free Space Maps
6,7:	Symbol Table Free Space Maps
8..:	Start of Used Symbol Table Space

DBMS LOGICAL UNITS FOR A USER

Table 6 shows the Logical Unit Table, local to a given user's task. Logical unit numbers 16-31 are reserved for DBMS usage. By way of example, say the page space for a relation is not in SMDB or LMDB (on the same physical machine) and must therefore be read in from disk. Once the logical address is determined, this table is searched, the physical disk address is calculated, and I/O issued against the affected logical unit.

TABLE 6. Logical Unit Table for a Task

	Start Logical Address	Physical File Size	Open Count
Entry 16			
Entry 17			
.	.	.	.
.	.	.	.
.	.	.	.
Entry 31			

STATE VARIABLE FOR A RELATION

Within a user's program, there are generally many places where Data Base references exist and the current "state" of access with respect to each relation must be retained. If, for example, a Data Base access exists in some isolated routine, the relation is "passed" to that routine (actually only the state variable is passed). Referring to Table 7, the state variable, per relation, contains several items.

Relation Error Status:

Contains the type and resultant error of Data Base access that was last done.

Relation Name:

Name of the Relation.

Owner Name:

Name of the owner

User Name:

ame of the user.

Current Tuple id:

Which tuple within the relation that was last accessed.

Current Attribute Number:

Which attribute within the tuple that was last accessed.

Function id:

Function last performed.

Line Number:

Pascal source line number where the Data Base access was called from.

Error Number:

Zero if no error; otherwise the number of the error for the function last performed.

Logical Unit Table Memory Address:

Memory location where the LU Table (see Table 6) exists.

Relation RTE Pointer:

Location of the main RTE for the relation (see Tables 3A-D).

User Access Rights:

Set of access restrictions applied to the specific user for the relation.

Workspace Tuple id:
Workspace Current LA:
Workspace Start:

Used by the DBMS for internal operations, dependant on the function performed.