A DESCRIPTION OF ADBMS

Version D2.0

by

Ernest Allen Hershey III
Richard L. Dissen
Paul W. Messink

ISDOS Working Paper No. 122

July 1975

ISDOS Research Project
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48104
(313) 763-3469
763-5329

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

1. INTRODUCTION

This working paper presents a description of a particular data base management system, ADBMS. The reader should be familiar with the concepts and data description language (DDL) used by the data base system, described in working paper No. 88, July 1975, entitled A DATA BASE MANAGEMENT SYSTEM FOR PSA BASED ON DBTG 71.

1.1 Purpose

This working paper is meant to serve several purposes. Its primary purpose is to give the reader a detailed understanding of how ADBMS operates. Secondly, it is meant to describe the logical structure of the data base, and of the data base object schema. Thirdly, it is meant to describe the data base control system (DBCS) routines as to their function, calling parameters, return codes, etc.*

1.2 Organization

Section 2 of this working paper gives a general overview of the complete data base management system. The sections following the overview describe in detail not only the logical and physical structure of the system, but also describe how the system accesses and updates the data base, and how the system performs the secondary functions of storage allocation and page management. The specific arrangement of the sections is as follows:

Section 2	ADBMS Overview
Section 3	Object Schema Control Blocks (DBTF)
Section 4	The Data Base, and Data Base Control Blocks (DBF)
Section 5	Data Base Storage Allocation
Section 6	Data Base Page Management
Section 7	The Data Base Control Systems (DBCS)
Section 8	Data Base Utility Programs

In addition, a glossary and an appendix are included at the end of the working paper to aid the reader in understanding the material presented.

2. ADBMS OVERIVEW

ADBMS is a data base management system, consisting of four parts:

(1) The data base (DBF) which consists of the data that is to be accessed.

(2) The data base tables (DBTF), otherwise known as the Object Schema. This is the logical description of the structure of the data base.

---

* Throughout this paper the term fullword and halfword are used as they apply to IBM 370 computers. In particular, fullword means a 32 bit word and a halfword means a 16 bit word. A byte is number of bits needed to store one character (8 bits in the case of the 370).

(3) The data base control system (DBCS), which consists of a collection of subroutines callable from FORTRAN. This is the actual programmed interface to the data base.

(4) The data base utility routines which are three stand-alone programs which aid the analyst in creating and maintaining a collection of data bases.

Section 2.1 of this overview describes the DBF, and the DBTF is described in section 2.2. Sections 2.3, 2.4, and 2.5 explain the DBCS, and section 2.6 describes the three utility programs.

2.1 The Data Base Table File (DBTF)

The DBTF contains the data base tables. The DBTF is generated by a program named DDLA, whose input is a data base description written in the Data Description Language (DDL).

The DBTF consists of two object schema tables (RECTAB and SETTAB), a character vector of DDL names (NAMES), and five fullwords of control information, as described in section 3.4.1.

2.1.1 The Record Table (RECTAB)

RECTAB can be viewed as a linear vector containing two types of object schema control blocks: Record Description Blocks (RDB) and Item Description Blocks (IDB). The logical structure of data in the DDL consists of a data base record and the data base items which belong to the record. There is an RDB for each record described in the DDL, followed by an IDB for each item belonging to that record.

The RDB and IDB are structured into fields, which contain pointers into NAMES, control information needed by the DBCS, and the lengths and displacements of other control areas and of physical data; generally any information needed to describe the logical structure of records and their associated items.

2.1.2 The Set Table (SETTAB)

SETTAB can also be seen as a linear vector of object schema control blocks; Set Description Blocks (SDB), Owner Description Blocks (ODB) and Member Description Blocks (MDB). These three types of control blocks describe the logical structure of the records in the data base. Each set described in the DDL will have an SDB located in the set tables. Following each SDB will be an ODB for each record type which is a legal owner of the set, followed by an MDB for each record type which is a legal member of the set.

The SDB, ODB, and MDB are also structured into fields. In addition to the NAMES pointers and control information are pointers to records in the data base, and pointers to other pointer areas in the data base itself. These pointers provide the DBCS with the information needed to add, update, and delete records from the data base.

2.1.3 The Names Vector (NAMES)

One way that objects in the schema (records, sets, etc.) are identified is by their DDL name. As the DBTF is being generated, each DDL name encountered is placed into the character array NAMES. The DBCS uses this vector when searching the data base for a particular object.

2.2 The Data Base File (DBF)

The information stored in the data base is placed by the DBCS into the DBF. The DBF consists of physical pages, usually defined at a computer installation to be equal in size to some unit of storage for that installation (i.e., track, page, etc.). For System/360-370, a data base page is the same size as a page of main memory, or 4096 bytes.

A data base page consists of two types of control blocks: a Page Header Information block(PHI), and Physical Record Header blocks (PRH).

2.2.1 The Page Header Information (PHI)

There is one PHI for every data base page, which appears at the beginning of each page. It contains control information needed by the DBCS to allocate space for new records.

2.2.2 The Physical Record Header (PRH)

One PRH preceeds each logical record on the page, and contains information used by the DBCS to identify the record type, if the record is currently being used for storage. If that record is not being used, it is referred to as a data base hole, and the PRH contains information needed by the DBCS to allocate that storage space or to combine the hole with other holes.

Following the PRH for a logical record are two additional blocks: the pointer area and the data area. The pointer area for a record (which is a member or an owner of a set) contains pointers to other records (which are owners or members of the same set). In this way, all the owners and members of a set can be logically connected.

The data area follows the pointer area. This area stores the data as it is added to the data base. Its physical structure cannot be generally described, since it is totally dependent upon the data description in the DDL. The logical structure is defined by the record type (RDB) of which the logical record is an occurrence.

2.2.3 Data Base Keys

Each logical record in the data base is uniquely identified by its data base key. This key is assigned by the DBCS when a record is entered into the data base, and is invariant as long as the record remains in the data base. The data base key is used by the DBCS to refer to

specific logical records. Its value is a function of the record's physical location in the data base. Thus it can be used as the value of a pointer in the control blocks and in the pointer areas.

2.3 Data Base Storage Allocation

When a data base is initialized, the entire area of each page (with the exception of the PHI) is availabe to the DBCS. When a record is entered into the data base, a block of storage is allocated by the DBCS, and no longer is availabe. If that record is subsequently deleted from the data base, the storage space is released, and is placed back into the pool of available storage. The allocation and deallocation of storage is totally under the control of the DBCS; it is transparent to the program calling the DBCS routines.

When a section of a data base page is being used for storage, it is referred to as a data base data record, or a logical record. If the block is not being used, it is called a hole, and is placed on a hole chain. When allocating storage, the DBCS uses the hole chain to provide the "best fit" of the new record into an availabe hole.

2.4 Data Base Page Management

A data base may range in size from 1 page to 99,999 pages. Since it is expensive (and often physically impossible) to have an entire, large data base in main memory at one time, the DBCS has a page management system which uses random data base page input and output routines to allow an efficient transfer of the data base between main memory and the DBF. If a record is needed that is not in main memory, the page on which that record appears is read into main memory. If a page needs to be removed from main memory to allow room for a new page, the DBCS uses an alogrithm to determine which page was used the longest time ago; it is probable that this page will least likely be needed soon in the future. If the page has been modified while in main memory, it is rewritten into the DBF before the new page is read in over it. The page management system is under the complete control of the DBCS, and is transparent to the program calling the DBCS routines. The calling program may assume that every record is available for use at any time.

2.5 The Data Base Control System (DBCS)

The DBCS is a collection of FORTRAN routines which interface with the user's program and with the ADBMS utility programs. They are divided into four groups, classified by function. The four groups are:

DBUSER
DBLOW
DBTAB
DBRAND

The DBUSER routines are the only routines that directly interface with the user's program. They contain comprehensive error checking to protect

the security of the data base against system errors and most user errors.

The DBLOW routines are the lower level routines used by DBUSER to access the data base; they also contain much of the program logic for the data base storage allocation system and for the data base page management system.

DBTAB is a collection of FORTRAN integer functions which return control block fields, and FORTRAN subroutines which update the control block fields. They are heavily used by DBUSER and DBLOW in referencing the data base tables.

DBRAND consists of random input/output routines used by DBUSER and DBLOW to transfer pages of the data base between main memory and the DBF.

In addition to these routines, a BLOCK DATA for the DBCS must be included by the user which specifies additional run-time control information for the routines.

2.6 ADBMS Utility Programs

There are three utility programs available for use with ADBMS. Each is a stand alone program, which calls routines in the DBCS.

The programs are:

(1) The Data Description Language Analyzer (DDLA)

This program generates the data base tables (DBTF) from a DDL. It also produces a FORTRAN BLOCK DATA source subprogram which is used by the DBCS. In addition, a DDL summary is produced for every record-type and set-type in the data base.

(2) The Data Base Initialization Program (DBIN)

This program uses the DBTF generated by DDLA to initialize a data base (DBF).

(3) The Data Base Summary Program (DBSM)

DBSM produces, for a populated data base, summary information on the sizes and percentage utilization of data base holes and records.

3. OBJECT SCHEMA CONTROL BLOCKS

The object schema is the computer's description of the data base which is generated from the user's description, the DDL. A program (DDLA) takes the DDL as input and generates the Data Base Table File (DBTF), which contains the object schema. The record table (RECTAB) contains control blocks for data base records and items. The set table (SETTAB) contains control blocks for sets, owner records, and member records,

while the character vector of names (NAMES) contains the DDL names for reference by the DBCS.

3.1 The Data Base Record Table (RECTAB)

The data base record table is made up of record description blocks (RDB) and item description blocks (IDB). There is one RDB for every record described by the DDL, and there is one IDB for each item associated with each record. For example, for the following lines included in a DDL:

```
RECORD    EMPL
ITEM      NAME     CHAR    30
ITEM      SSNO     INTEG   31
ITEM      TITLE    INTEG    7
ITEM      PAYRTE   REAL    31
```

there will be one RDB and four IDB.

3.1.1 Record Description Blocks (RDB)

An RDB consists of six contiguous fullwords of information used to describe record types in the data base. (The terms "RDB" and "record types" are used interchanageably.) The following diagram describes the physical structure of the RDB:

0	RDBNMP	RDBNML
1	RDBCUR	
2	RDBITM	RDBMLR
3	RDBPAL	RDBDAL
4	RDBOWN	RDBMEM
5	RDBBTS	RDBDIS

Fullword

There are ten halfword integers and one fullword integer of storage contained in the RDB. The following table, Table 3.0, describes the meaning of each storage location. The "F" and "S" columns indicate whether there is a function (F) or a subroutine (S) in DBTAB which can be used to reference or change the appropriate storage location. See section 7 of this working paper for a full explanation of the use of these routines.

Table 3.0

FIELD NAME	SIZE	F	S	DESCRIPTION
RDBNMP	HALFWORD	X		RECORD NAME POINTER (BYTES)
RDBNML	HALFWORD	X		RECORD NAME LENGTH (BYTES)
RDBCUR	FULLWORD	X	X	DATA BASE KEY OF CURRENT OF RECORD TYPE
RDBITM	HALFWORD	X		NUMBER OF ITEMS IN RECORD TYPE
RDBMLR	HALFWORD	X		MAXIMUM LENGTH OF RECORD (WORDS)
RDBPAL	HALFWORD			POINTER AREA LENGTH (WORDS)
RDBDAL	HALFWORD	X		DATA AREA LENGTH (WORDS)
RDBOWN	HALFWORD	X		NUMBER OF SETS OF WHICH THIS RECORD IS AN OWNER
RDBMEM	HALFWORD	X		NUMBER OF SETS OF WHICH THIS RECORD IS A MEMBER
RDBBTS	HALFWORD	X		RECORD BITS
RDBDIS	HALFWORD	X		DISPLACEMENT OF DATA AREA (WORDS)

RDBNMP is the pointer into the character array NAMES where all the DDL names are stored, and RDBNML is the name length. The data base key of the current record of this RDB is stored in RDBCUR. The current record may be changed using subroutine RDSCUR. The pointer area length and data area length for this record type are stored in RDBPAL and RDBDAL. RDBBTS stores the record bits for the record type. The record bits classify the record in the following way:

RDBBTS

1	Fixed Length Record
2	Variable Length Record
4	System Record

Every record is either of fixed length or variable length. The system record has an RDBBTS value of 5, since it is of fixed length and it is the system record.

The displacement of the data area in the data base data record is kept in RDBDIS.

To reference an RDB, the data base routines use an RDB pointer. This pointer contains the displacement into RECTAB where the first halfword begins. In any data base an RDB pointer specifies a unique RDB.

3.1.2 Item Description Blocks (IDB)

In RECTAB there is an IDB for every item associated with every record in the DDL. The IDB consists of five contiguous fullwords of storage. These five fullwords consist of nine halfword integer locations and one unused halfword. The following diagram describes the physical structure of the IDB.

0	IDBNMP	IDBNML
1	IDBTYP	IDBBTS
2	IDBMLI	IDBMVD
3	IDBDDP	IDBDSP
4	IDBTMV	- not used -

The next table describes the meaning of each field:

FIELD NAME	SIZE	F	S	DESCRIPTION
IDBNMP	HALFWORD			ITEM NAME POINTER (BYTES)
IDBNML	HALFWORD			ITEM NAME LENGTH (BYTES)
IDBTYP	HALFWORD	X		ITEM TYPE
IDBBTS	HALFWORD	X		ITEM BITS
IDBMLI	HALFWORD	X		MAXIMUM LENGTH OF ITEM (WORDS)
IDBMVD	HALFWORD	X		MAXIMUM VALUE OF DEPENDING ITEM (WORDS)
IDBDDP	HALFWORD	X		DISPLACEMENT OF DEPENDING ITEM (WORDS)
IDBDSP	HALFWORD	X		DISPLACEMENT OF ITEM (WORDS)
IDBTMV	HALFWORD	X		ITEM TYPE MODIFIER VALUE

IDBTYP identifies the item type, according to the following table:

IDBTYP	TYPE
1	INTEGER
2	INTEGER
4	REAL
8	REAL
16	BINARY
32	DATA BASE KEY
64	LOGICAL
128	CHARACTER

The next table describes the values of IDBBTS, and their meanings.

IDBBTS	ITEM CLASS
1	SINGLE ITEM
2	FIXED LENGTH ITEM
4	VARIABLE LENGTH ITEM
8	DEPENDED ON ITEM

In the current implementation, an item may be classified into only one of the above four item classes, the exception being that a single item may be a depended on item also (IDBBTS=9).

IDBMLI is the value for the maximum length of the item, in words. IDBDDP is the displacement in the record data area of the depending item, and IDBDSP is the displacement in the data area of the item itself. IDBTMV is the type modifier value given the item in the DDL. For example, for the DDL statement

```
ITEM NAME CHARACTER 30
```

the type modifier value is 30.

Each IDB in the data base can be uniquely referenced using an IDB pointer. This value of the pointer is the displacement (in words) into RECTAB of the first halfword of the IDB.

DEPENDING ITEMS

A repeating item may be specified for a record by specifiying in the DDL the depending on item, and a maximum value of the depending item. If an item is a repeating item, the IDBMVD and IDBDDP fields of the IDB contain the appropriate data. If the item is a single item, these two fields are ignored.

3.2 The Data Base Set Table (SETTAB)

The data base set table is made up of set description blocks (SDB), owner description blocks (ODB) and member description blocks (MDB). There is one SDB for every set in the DDL. For each SDB there is one ODB for every record type which is a legal owner of the set-type, and one MDB for every record type which is a legal member of the set-type. For example, for the following lines included in a DDL:

```
SET ALLEMP SORTED SSNO
OWNER SYSTEM
MEMBER EMPL
```

there will be one SDB, one ODB, and one MDB.

3.2.1 Set Description Blocks (SDB)

A set type is defined by an SDB, which consists of seven contiguous fullwords of storage. These seven fullwords contain ten halfword integer fields and two fullword fields. The following diagram illustrates the physical structure of the SDB.

0	SDBNMP	SDBNML
1	SDBCRO	
2	SDBCRM	
3	SDBSNP	SDBSNL
4	SDBSKM	SDBOBT
5	SDBSKT	SDBSKL
6	SDBNMO	SDBNMM

The next table describes the meaning of each SDB field.

FIELD NAME	SIZE	F	S	DESCRIPTION
SDBNMP	HALFWORD			SET NAME POINTER (BYTES)
SDBNML	HALFWORD			SET NAME LENGTH (BYTES)
SDBCRO	FULLWORD	X	X	DATA BASE KEY OF CURRENT OWNER OF THE SET
SDBCRM	FULLWORD	X	X	DATA BASE KEY OF CURRENT MEMBER OF THE SET
SDBSNP	HALFWORD	X		SORT KEY NAME POINTER (BYTES)
SDBSNL	HALFWORD	X		SORT KEY NAME LENGTH (BYTES)
SDBSKM	HALFWORD	X		SORT KEY TYPE MODIFIER (see IDBTMV)

(cont.)

FIELD NAME	SIZE	F	S	DESCRIPTION
SDBOBT	HALFWORD	X		SET ORDER BITS (BYTES)
SDBSKT	HALFWORD	X		SORT KEY TYPE (see IDBTYP)
SDBSKL	HALFWORD	X		SORT KEY LENGTH (WORDS)
SDBNMO	HALFWORD			NUMBER OF LEGAL OWNER RECORD TYPES
SDBNMM	HALFWORD			NUMBER OF LEGAL MEMBER RECORD TYPES

The data base keys of the current owner and current member of the set are stored in SDBCRO and SDBCRM. They may be changed using routines SDSCRO and SDSCRM, which are described in section 7 of this working paper.

The order in which the members are sorted in and retrieved from the set is determined by SDBOBT, according to the following table.

SDBOBT	ORDER TYPE
1	FIFO
2	LIFO
4	NEXT
8	PRIOR
16	IMMATERIAL
32	SORTED ON...

Each SDB in the data base can be uniquely referenced using an SDB pointer. The value of the pointer is the displacement into SETTAB of the first halfword in the SDB.

SORTED SETS

A set is sorted when it has an SDBOBT value of 32. SDBSNP and SDBSNL reference the name of the item on which the set is ordered, known as the "sort key." SDBSKM has the type modifier value for the sort key, which should be the same as IDBTMV for the sort key item. The sort key type and length are specified by SDBSKT and SDBSKL, and should also be the same as the type and length of the sort key item.

3.2.2 Owner Description Blocks (ODB)

Each owner record type for a set is defined by an ODB, which consists of two contiguous fullwords of storage. In these two fullwords are three halfword integer fields and one unused halfword field. The following diagram shows the physical structure for an ODB.

0	ODBNMP	ODBNML
1		ODBDSP

The next table describes the meaning of each ODB field.

FIELD NAME	SIZE	F	S	DESCRIPTION
ODBNMP	HALFWORD			OWNER NAME POINTER (BYTES)
ODBNML	HALFWORD			OWNER NAME LENGTH (BYTES)
ODBDSP	HALFWORD	X		DISPLACEMENT OF OWNER POINTERS (WORDS)

ODBDSP is the displacement of the owner pointers in the data area of the owner record.

Each ODB in the data base can be uniquely referenced using an ODB pointer. The value of this pointer is the displacement into SETTAB of the first halfword in the ODB.

3.2.3 Member Description Blocks (MDB)

Each member record type for a set is defined by an MDB, which consists of two contiguous fullwords of storage. In these two fullwords are three halfword integer fields and one unused halfword. The following diagram shows the physical structure for an MDB.

0	MDBNMP	MDBNML
1		MDBDSP

The next table describes the meaning of each MDB field.

FIELD NAME	SIZE	F	S	DESCRIPTION
MDBNMP	HALFWORD			MEMBER NAME POINTER (BYTES)
MDBNML	HALFWORD			MEMBER NAME LENGTH (BYTES)
MDBDSP	HALFWORD	X		DISPLACEMENT OF MEMBER POINTERS (WORDS)

MDBDSP is the displacement of the member pointers in the data area of the member record.

Each MDB in the data base can be uniquely referenced using an MDB pointer. The value of this pointer is the displacement into SETTAB of the first halfword in the MDB.

3.3 The Character Array NAMES

The third table that is generated from a DDL is a character array where the names of all the DDL records, items, and sets are stored. The array is used by the data base routines when searching the data base for a particular record type, set type, etc. The index into the table is not kept in one place; rather, each RDB, IDB, SDB, MDB, and ODB has a name pointer field and a name length field. The name pointer is the displacement (in bytes) into NAMES where the name begins, and the name length field has the value of the name length (also in bytes). (In the current implementaiton, all name lengths are by default six characters. If a DDL name is less than six characters, it is padded on the right with blanks.)

The following table illustrates the name types that are stored in NAMES, and where the name pointer and length fields are stored.

NAME TYPE	NAME POINTER	NAME LENGTH	WHERE STORED
RECORD	RDBNMP	RDBNML	RDB
ITEM	IDBNMP	IDBNML	IDB
SET	SDBNMP	SDBNML	SDB
SORT KEY	SDBSNP	SDBSNL	SDB
OWNER RECORD	ODBNMP	ODBNML	ODB
MEMBER RECORD	MDBNMP	MDBNML	MDB

It should be noted that an item is not uniquely identified by its DDL name, since two distinct record types may have items with the same name. When searching RECTAB for an item name, the record type in which the item appears must also be specified, to guarantee that the correct item has been found.

Since record and set names are unique in the DDL, they can be uniquely identified by their DDL name.

3.4 DBTF structure as output from DDLA

This section will describe the physical order of the data base control blocks as written into the DBTF by DDLA.

There are four logical records of table information:

1. The object schema parameters
2. RECTAB
3. SETTAB
4. NAMES

Each logical record is written into the DBTF using a single, unformatted FORTRAN WRITE statement. The DBTF is read into main memory each time the data base is opened using four unformatted FORTRAN READ statements.

3.4.1 The Object Schema Parameters

There are five parameters which are contained in the first line of the DBTF. They are NPAGES, the number of pages in the data base (taken from the NPAGES statement in the DDL); PAGSIZ, the machine page size in words (system dependent); RECLEN, the length of RECTAB (in words); SETLEN, the length of SETTAB (in words); and NAMLEN, the length of NAMES (in words).

3.4.2 RECTAB

The RDB and IDB are stored in RECTAB, in the order which they appear in the DDL. The first control block defines the SYSTEM record. After the SYSTEM RDB is the RDB for the first record type in the DDL, followed by the IDB for the items in that reocrd type. The IDB appear in the order specified in the DDL. The next control block is the RDB for the next record type, followed by its IDB. This pattern continues for each RDB and IDB in the data base. The following diagram illustrates the structure of RECTAB.

RDB	RDB	IDB	...	IDB	RDB	IDB	...	IDB	RDB	....

3.4.3 SETTAB

The set table contains all the SDB, ODB, and MDB for the data base, in a manner similar to RECTAB. The first control block in SETTAB is the SDB for the first set specified in the DDL. Following the SDB is one or more ODB (depending on the value of SDBNMO) and one or more MDB (depending on the value of SDBNMM). Following the last MDB is the SDB for the next set, along with its ODB and MDB.

For the follwoing statements found in a DDL:

```
SET  ALLEMP  FIFO
OWNER  SYSTEM
MEMBER  EMPL
MEMBER MANAGR
```

This section of SETTAB would be generated.

SDB (ALLEMP)	ODB (SYSTEM)	MDB (EMPL)	MDB (MANAGR)

3.4.4 Names

The names of all the objects described in the DDL are stored in the array NAMES (see section 3.3). The first name stored will always be "SYSTEM." Following this are the names of the DDL records, items, and sets. The names are stored in the order found in the DDL, except when an item is described which has the same name as a previous item. In this case, only the first instance of the name is stored, and IDBNMP for the duplicate name will point to the first occurrence of the name.

In the current implementation, if the DDL name is less than six characters in length, the name is padded on the right to insure the correct name length.

4. DATA BASE CONTROL BLOCKS

As information is stored in the data base it is structured by the DBCS into physical records, and is stored in the Data Base File (DBF). This section describes the structure of the physical records which appear in the DBF.

4.1 The Page Header Information (PHI)

The DBF consists of 1 to 99,999 physical pages (defined on System/360-370 to be 4096 bytes each). The first two fullwords on each page are reserved for the PHI, which contains four halfword integer fields of page identification and control information. The following diagram illustrates the physical structure of the PHI.

0	PHIPNM	PHIMHS
1	PHIHCH	PHIBTS

The next table describes the meaning of each PHI field.

FIELD NAME	SIZE	F	S	DESCRIPTION
PHIPNM	HALFWORD	X		PAGE NUMBER
PHIMHS	HALFWORD	X	X	MAXIMUM HOLE SIZE ON THE PAGE
PHIHCH	HALFWORD	X	X	HOLE CHAIN HEADER
PHIBTS	HALFWORD	X	X	PAGE BITS

PHIPNM contains the page number for the page. The pages are numbered sequentially beginning at page 1.

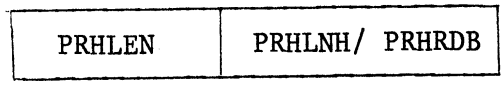
PHIMHS and PHIHCH contain information used by the DBCS in allocating and releasing storage space on the page. See section 5 for a detailed

description of the data base space allocation system.

The PHIBTS field is a switch used by the DBCS when modifying the data base. If a page is in main memory and it has a PHIBTS value of 1, it means that the page has been modified in some way, and must be re-written into the DBF before closing the data base page. See section 6 for an explanation of the data base page management system.

4.2 Physical Record Header (PRH)

The PRH consists of one fullword of storage which preceeds every data base data record and every data base hole. The two halfword integer fields contain control information for that record or hole. The following diagram illustrates the structure of the PRH.



The next table describes the meaning of each field in the PRH.

FIELD NAME	SIZE	F	S	DESCRIPTION
PRHLEN	HALFWORD	X	X	LENGTH OF BLOCK
PRHLNH	HALFWORD	X	X	LINK TO NEXT HOLE (IF BLOCK IS DATA BASE HOLE)
PRHRDB	HALFWORD	X	X	RDB POINTER (IF BLOCK IS A DATA BASE DATA RECORD)

The PRHLEN field contains the length of the block, excluding the PRH. If the block is a data base hole, PRHLEN gives the size of the hole, and PRHLNH is the link to the next hole on the hole chain (see section 5). If the block is a logical record, PRHLEN is the block length, including the pointer area and the data area. PRHRDB is the pointer into RECTAB of the RDB (record type) which describes this record.

4.2.1 Data Base Holes

Unallocated storage space on a data base page is called a hole. The holes on a page are structured in such a way that the DBCS can find and allocate storage in a fairly efficient manner. A detailed explanation of the method in which the DBCS allocate and release storage is given in section 5 of this working paper.

4.2.2 Data Base Data Records

The DBCS stores the data base information in the data base data records (also called "data base logical records"). The length of the record is a function of three things:

1. The number and length of items in the record type which this record is an occurrence of.

2. The number of sets the record type is an owner of.

3. The number of sets the record type is a member of.

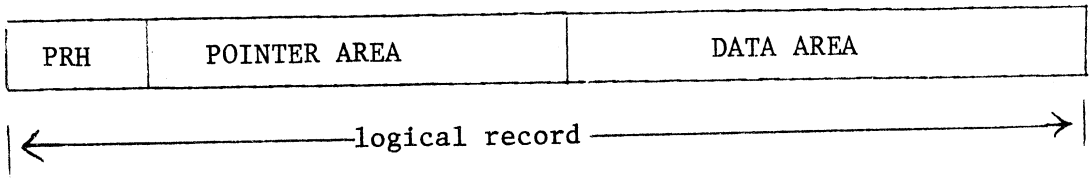
The structure of a logical record is:

1. The PRH

2. The pointer area

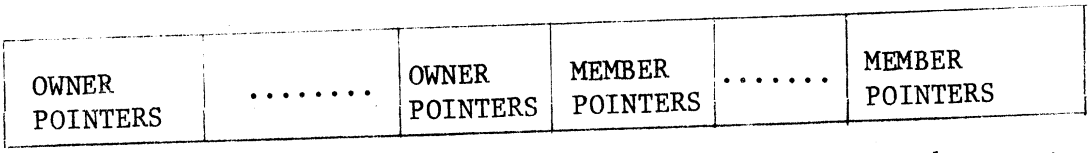
3. The data area

This structure is shown graphically by the following figure.



4.2.2.1 The Pointer Area

The pointer area for a data base record contains owner pointers and member pointers. A physical data record will have one set of owner pointers for every set of which the associated data base record type is a legal owner, and have one set of member pointers for every set of which the associated data base record type is a legal member. Each set of pointers (owner and member) uses three fullwords of storage. A more detailed diagram of the pointer area looks like this:



The owner pointers always preceed the member pointers. The exact number of owner and member pointers can be found for a record type in the RDB, in fields RDBOWN and RDBMEM. The length of the pointer area for a record type may be found also in the RDB, or may be calculated using the formula 3 x (RDBOWN + RDBMEM) since each set of pointers uses three fullwords.

4.2.2.1.1 Owner Pointers

An instance of a record type will have owner pointers for each set of which the record type is a legal owner. The contents of the owner pointer area is the data base keys for the first member and last member of the set, as well as the count of the number of members in the set.

This is shown as follows:

	0	1	2
OWNER POINTERS	DATA BASE KEY OF FIRST MEMBER	DATA BASE KEY OF LAST MEMBER	NUMBER OF MEMBERS

4.2.2.1.2 Member Pointers

An instance of a record type will have member pointers for each set of which the record type is a legal member. The contents of the member pointer area is the data base keys of the previous member of the set, the owner of the set, and the next member of the set. Graphically:

	0	1	2
MEMBER POINTERS	DATA BASE KEY OF PREVIOUS MEMBER	DATA BASE KEY OF OWNER RECORD	DATA BASE KEY OF NEXT MEMBER

4.2.2.1.3 Data Base Keys

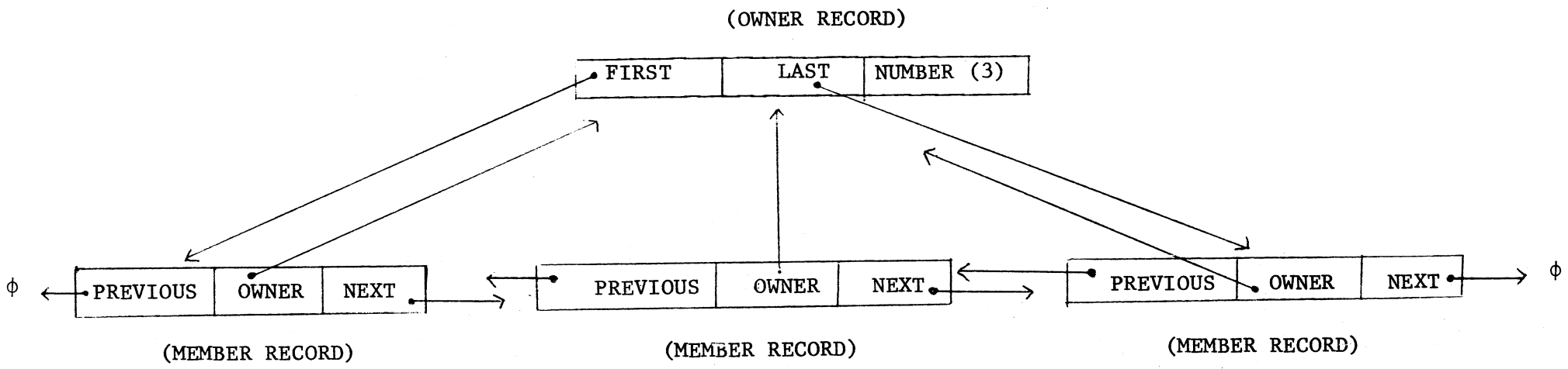
Every data base data record has a unique identifier called a data base key. The data base key is a fullword of storage, consisting of two halfword integer fields.

PAGE NUMBER	DISPLACEMENT ON PAGE

The first halfword of the data base key contains the page number of the data base page which the logical record is stored on. The second halfword of the key contains the displacement (in words) of the logical record on the page. Using these two fields the DBCS can locate and identify any record in the data base.

4.2.2.1.4 Links Between Owner Records and Member Records

The data base keys in the owner and member pointers form doubly linked lists of records: the owner pointers contain pointers to the heads of lists, and for each owner there are member records that form the elements on the list. A set with three members would be chained together in the manner shown in the following figure.



Note that the previous member of the first member of the set and the next member of the last member of the set are set to null. This means that rather than containing a data base key, those two fields have a value of zero.

The members associated with a particular owner of a set are entered onto the linked list in a specific order, as specified in the DDL. The DBCS checks the set order bits (SDBOBT) and adds new members in the appropriate place on the list.

4.2.2.2 The Data Area

The information stored in the data base is kept in the data area. Each record in the data base (except for the SYSTEM record) should have one or more items associated with it. The length of each item is specified in the DDL, and room is reserved in the data area for each item, in the order that the items appear in the DDL.

It is impossible to show the general structure of the data area, since it is determined totally from the DDL. The data in each record is located and retrieved using the IDB for each item in the record. The following table is a summary of the IDB fields which reference the data area in some way.

FIELD NAME	DESCRIPTION
IDBTYP	Item Type (Integer, Real, Binary, Data Base Key, Logical, Character)
IDBBTS	Item Bits (Single, Fixed, Variable, Depended on)
IDBMLI	Maximum Length of Item
IDBMVD	Maximum Value of Depending Item
IDBDDP	Displacement of Depending Item in Data Area
IDBDSP	Displacement of Item in Data Area
IDBTMV	Type Modifier Value

In addition, several fields in the record's RDB are used to locate and reference the data in the data area. These RDB fields are:

FIELD NAME	DESCRIPTION
RDBITM	Number of Items
RDBMLR	Mamimum Length of the Record
RDBPAL	Pointer Area Length
RDBDAL	Data Area Length
RDBBTS	Record Bits
RDBDIS	Displacement of Data Area in the physical record

If an item is a repeating item, the DBCS will reserve storage for the maximum number of occurrences of that item (see the IDBMVD field).

5. DATA BASE STORAGE ALLOCATION

The DBCS contains a data base storage allocation/deallocation system which is used to create and delete logical records in the data base. When a block of storage is allocated for use by the DBCS, it is

called a data base data record, or logical record. When the storage is released, it is available for re-use by the DBCS, and is called a data base hole. Each hole on a data base page is a member of the hole chain for that page. The hole chain is a linked list of holes; its management is under the complete control of the DBCS.

There is no indication in the data base of whether a particular block on a page is a hole or a logical record. The DBCS has an implicit knowledge of whether a storage block is a hole or not in this way: a logical record should not appear as a member on the hole chain. Any storage block on the hole chain is assumed to be a data base hole. Conversely, any storage block that is pointed to by a data base control block (i.e., by an RDB or SDB) is assumed to be a logical record, and not a hole.

This section describes the structure of hole chain, how it is used by the DBCS, and how storage space is allocated and released.

5.1 The Hole Chain

Each data base page has a hole chain, which is independant of the hole chain of any other page. Two of the four halfwords of the PHI are used by the DBCS in storage space mangement as follows:

FIELD NAME	DESCRIPTION
PHIMHS	Maximum Hole Size on this Page
PHIHCH	Hole Chain Header

The PHIHCH field contains the displacement on the page of the PRH of the first hole logically on the page. For a data base hole, the first PRH field is the length of the hole (PRHLEN) and the second field is a link to the next hole on the chain. The last hole on the chain has a link which is set to null (the value of the link is zero). The holes are placed on the hole chain by the DBCS in ascending order of size. If there are no holes on a particular page, PHIHCH is set to null, and PHIMHS is set to zero.

5.2 Storage Allocation

When the DBCS searches a page for some available storage to allocate, it first checks PHIMHS to see if there is a contiguous block of space on the page with a size equal to or greater than the desired block size. If not, another page will be checked.

If the PHIMHS field indicates that the space is available, the DBCS begins to search the hole chain. The search ends when the first block is found whose size is greater than or equal to the desired block size. Since the chain is ordered by increasing size, the block chosen by the DBCS will be the "best fit;" i.e., it will be the smallest block on the page that is big enough to hold the desired allocation.

The block found by the DBCS is taken off the hole chain. If the size of the block exceeds the desired size by more than two fullwords, the excess storage is separated from the allocated block, and is returned to the hole chain, in the proper location. The allocated block is then assigned a data base key, and is given to the DBCS to be used as a logical record.

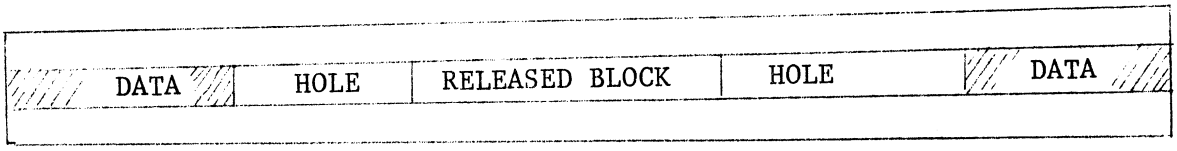
If the size of the block found by the DBCS is greater than the desired size by two fullwords or less, the excess one or two words are included in the allocated block, and is treated as "padding." This is done because the excess words are too small to be used elsewhere and should not be put back onto the hole chain. The PRHLEN field for the block contains the value for the length of the block, including padding. When the block becomes an instance of a record type, the RDBMLR field of the record's RDB will contain the length of the record, excluding any padding that may be present. It is important to realize that the values of the two fields will not always be equal.

5.3 Storage Deallocation

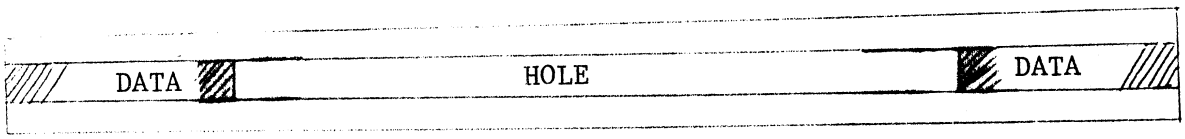
When a logical record is deleted from the data base, the storage space is deallocated (or released), and becomes available again to the DBCS for reuse.

Before a storage block is returned to the hole chain, the DBCS checks the entire chain to see if there is a hole on the chain that is stored physically before or after the released block. If a contiguous hole is found, it is taken off the chain and combined with the released block, forming a larger hole. When the entire chain has been searched, the hole is put onto the hole chain in its proper location.

As an illustration, take the following example of a section from a data base page:



After combining, the same section looks like this:

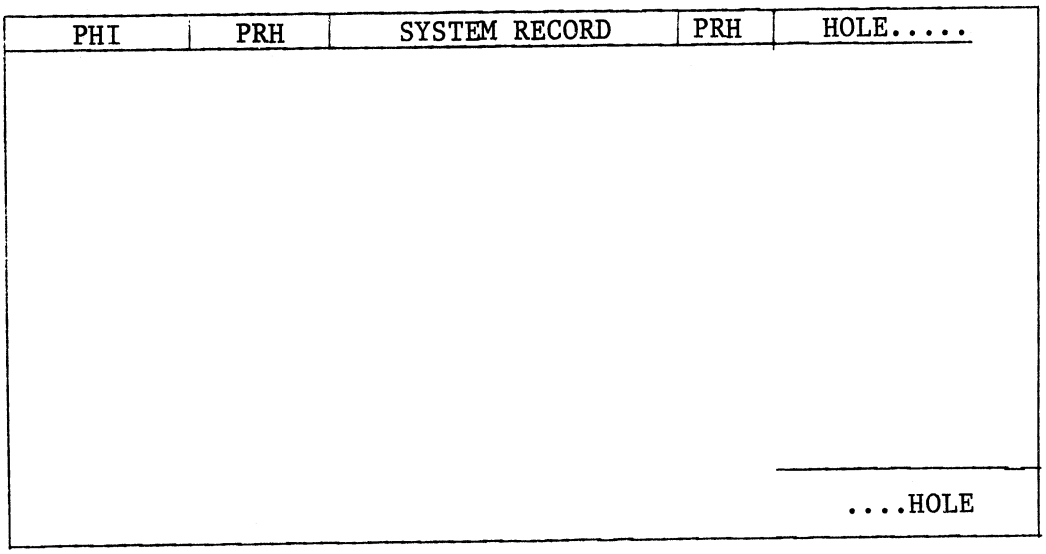


6. DATA BASE PAGE MANAGEMENT

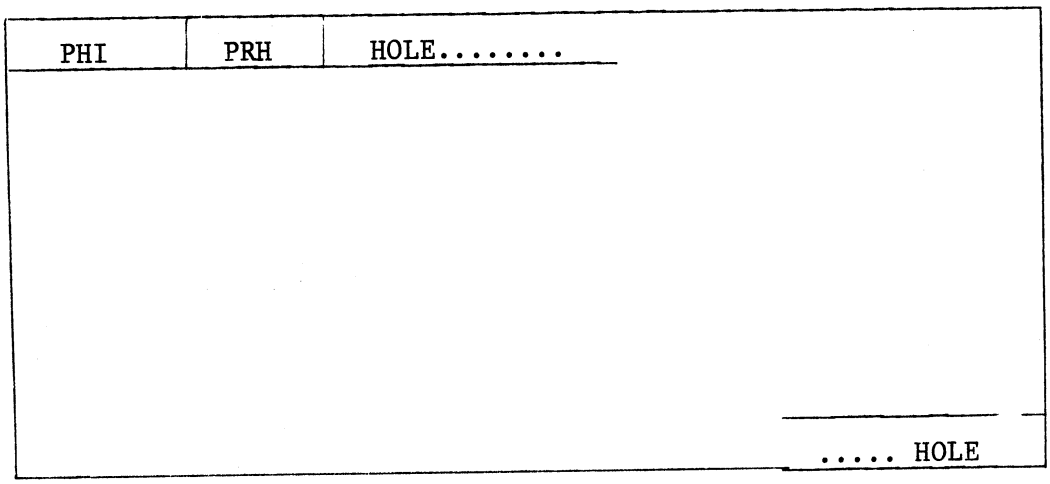
Sections 3 and 4 of this working paper discussed the control blocks necessary to reference and update the data base pages. Section 5 explained the data base storage allocation/deallocation algorithm. This section will use the three previous sections to describe the total data base page structure. It will also describe the page management system used by the DBCS.

6.1 The DBF structure

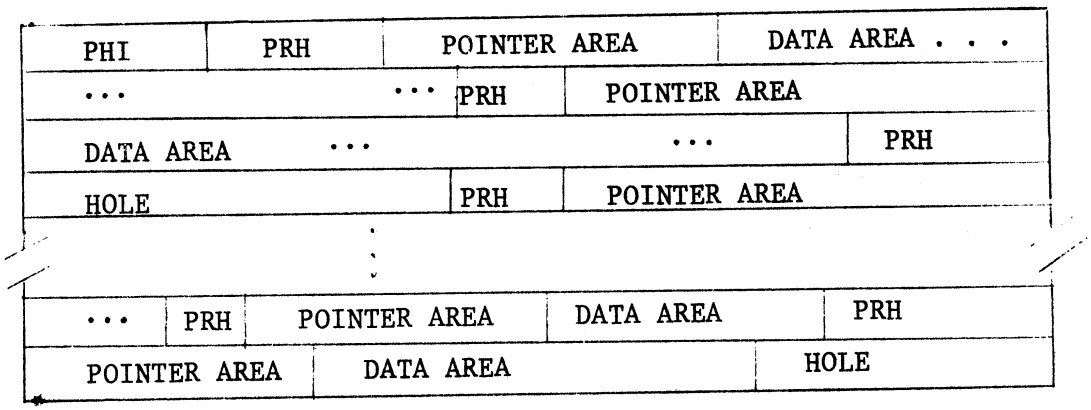
As described in ISDOS working paper No. 88, the second step in creating a data base is to intialize the DBF using the DBTF. The result of this operation is an initialized data base; that is, the DBF contains only one logical record, an occurrencd of the SYSTEM record. The first page of the data base then looks like the following diagram.



Each succeeding page in the data base looks like the next diagram.



Once the data base has been populated, the holes are allocated as logical records. If some records are then deleted, the data base pages begin to have holes imbedded between records. The following diagram might be a typical page from this type of data base.



6.2 DBCS Page Management System

Many computer installations limit the number of pages of main memory that a program may use at any time. Since the size of a data base can be many times this limit, the DBCS contains a set of routines which will control which pages in the data base are kept in main memory. If a page is needed that is not in main memory at that time, the DBCS will go to the DBF and read in the page. This action is totally under the control of the DBCS. The user program may assume that every logical record in the data base is available to it at any time.

6.2.1 The DBCS Random I/O Routines

The basis of the page management system is a set of random input/output routines contained in the DBCS. These routines can read into main memory selected pages of the data base, and write them out again into the DBF, in their proper sequence. (Remember that the pages in the DBF are numbered sequentially.) Section 7.4 describes these routines.

6.2.2 The Page Buffer (PAGE)

The data base pages, when in main memory, are kept in a single, one dimensional integer array called PAGE. PAGE is dimensioned to a value equal to the page size (in words) multiplied by the maximum number of pages allowed in main memory at one time, MPICOR.

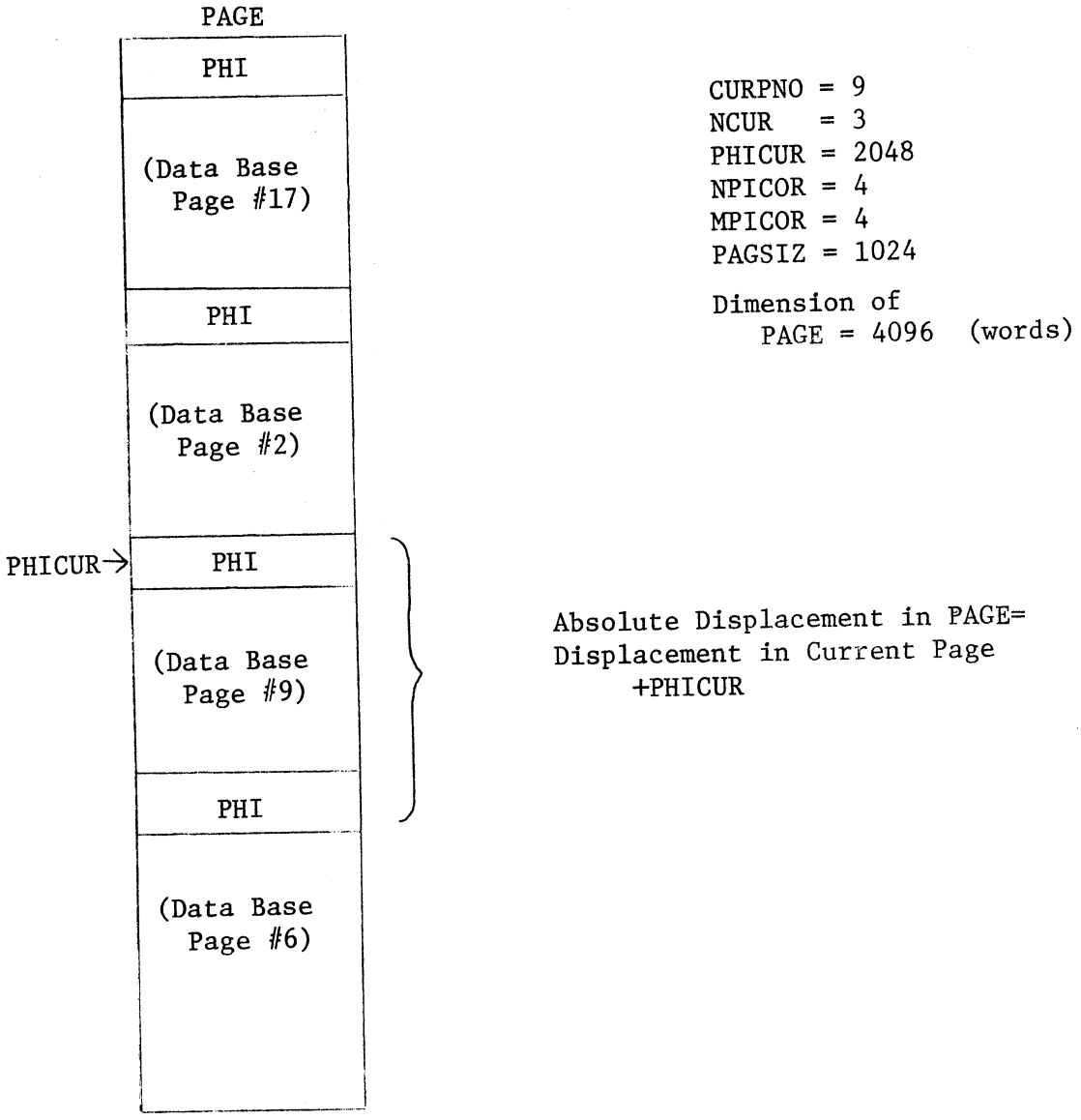
6.2.3 The Current Page

The DBCS flags one of the pages in main memory as the "current page". The page number of the current page is kept in an integer fullword named CURPNO. Several DBCS routines reference the current page rather than specify one particular page.

Associated with CURPNO is a fullword integer variable, PHICUR. PHICUR is a pointer to the PHI of the current page. Because the current page may be in one of several places in PAGE, a displacement into the current page is not necessarily the same as the displacement of the same location in the PAGE vector.

Finding the hole chain header of the current page, for example, would entail adding the displacement of the PHIHCH field (the displacement is 2 words) to the value of PHICUR, taking the left halfword of the location of the resulting absolute displacement.

The following diagram gives an example of a data base at a particular point of execution of a user program. MPICOR for this example is 4 pages, and PAGSIZ is 1024 (1024 words per page). This implies that PAGE is dimensioned to 4096. There are four data base pages in main memory: pages 17, 2, 9, and 6. NPICOR, the number of data base pages in main memory has a value of 4. Page 9 is the current page, which is sequentially the third page in main memory. NCUR, the sequence number of the current page in main memory, is then 3. PHICUR, the pointer into PAGE of the PHI for CURPNO, is 2048. (This is numerically equal to (NCUR-1) *PAGSIZ.)



When the current page changes, CURPNO and NCUR are changed, and PHICUR is recalculated. This means that the DBCS can always reference the correct location given the displacement on the current page.

6.2.4 Reading a New Page from the DBF into Main Memory

There is a particular algorithm used to read a page into main memory from the data base. The DBCS first checks to see if NPICOR is less than MPICOR. If so, there is still room in PAGE for a new data base page, and the page is read from the DBF into main memory. NPICOR is incremented by 1, and the new page becomes the current page.

If there is no room in main memory, one page currently in main memory must be selected to be removed. To do this, the DBCS utilizes a vector named PREF and a counter named NDBKF. PREF must be dimensioned to be at least the value of MPICOR. Each time the current page is reset, NDBKF is incremented, and stored in the location in PREF that is associated with the page sequence number in main memory (i.e., PREF(NCUR)). In this way, the page in main memory which corresponds to the lowest value in PREF is the page that was least recently referenced. This is the page that the DBCS selects to be removed. Since all the other pages in main memory have been referenced more recently, they normally have a greater chance of being referenced than the one being removed.

The DBCS checks the page bits of the page to be removed. If the value of the page bits is 1, the page is rewritten into the DBF, and the new page is read into main memory, replacing the old page. The current page is set to the page number of the new page.

If the page bits value of the old page is zero, it means that the old page was not modified since being read in; therefore it does not need to be written out into the DBF. The new page is read in, replaces the old page, and becomes the current page.

6.2.5 Setting the Current Page

There are two times that the current page is set by the DBCS. The first, reading in a new page, is described in the above paragraphs. The second time is when routine DBKFND is called. This routine accepts a data base key as input and returns the record's PRH pointer. DBKFND checks the page number of the input key; if the page is in main memory it is made the current page and the corresponding location in PREF is updated.

If the page is not in main memory, DBKFND uses the DBCS random I/O routines to read the page from the DBF, as described in section 6.2.4.

Although no other routines directly set the current page, most higher level DBCS routines reference DBFND during their execution.

6.2.6 Modifying the Current Page

The only page that can be modified by the DBCS is the current page. When modified, the page bits for the current page must be set to 1. When a page is replaced by a new page from the DBF or when the data base is closed, the page bits are checked; when they contain a value of 1, the page is rewritten into the DBF.

7. THE DATA BASE CONTROL SYSTEM (DBCS)

The DBCS is a set of FORTRAN routines which are used by the user's program to access a data base. The routines are divided into the four following groups, classified roughly by function.

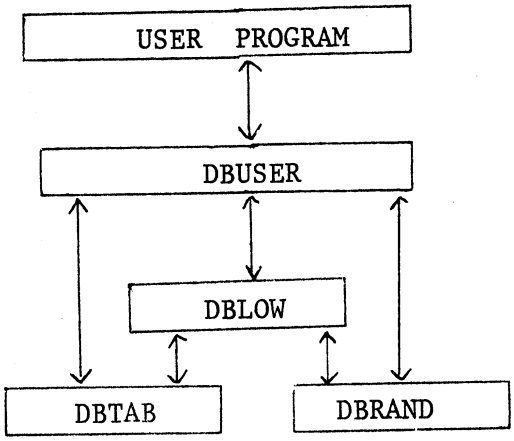
DBUSER - These routines are the highest level routines. They are the only routines that interface directly with the user's program.

DBLOW - These are the low level routines used by DBUSER. They do most of the actual "work" of the DBCS.

DBTAB - These routines are used by DBUSER and DBLOW to access the data base tables and control blocks.

DBRAND - These are the DBCS random I/O routines, which are used to open, read from, write onto, rewrite onto, and close the DBF.

The user program interfaces (or references) only DBUSER. The "hierarchy" of the DBCS routine groups is as follows:



7.1 DBUSER

The routines contained in DBUSER directly interface with the user's program. The routines give the user's program control over what data base is to be accessed, what information is to be stored in which sets, what the current status of the currency indicators are, etc. The routines can add and delete records in the data base, they can store and retrieve data fields, they can find members of a set, and check set ownership and membership.

The user's program does not need to do anything to insure the correct data base page is accessible, to locate available data base holes for record storage, to determine the physical location of a record or an item, etc. All these function are taken care of by DBUSER.

7.1.1 DBUSER Error Control and Data Base Security

Built into DBUSER is an error detection and notification system. Each routine in DBUSER begins by checking to make sure the data base is open. If so, a counter called LEVEL is incremented. LEVEL is set to 0 when the data base is opened; it is incremented each time a DBUSER routine is entered and decremented each time it exits. In this way, LEVEL keeps a count of the current "level" of nesting. When an error occurs, it is helpful to know the value of LEVEL, in order to trace which DBUSER routines called other DBUSER routines.

Each possible error encountered in DBUSER is assigned an error number (see ISDOS Working Paper No.88), which is returned to the calling program in the return parameter IERR. In addition, the error number, the routine name, and the value of LEVEL is printed out each time an error is detected in DBUSER.

Most of the errors that occur are due to invalid set types, record types, owners, or members being specified, or when currency indicators are not set. When an error is found, the execution of that routine is halted, the error information described above is printed, and an immediate return is made. Through the DBCS error checking facility the security of the data base is protected against system errors and against most user errors.

7.1.2 DBUSER Subprogram Descriptions

Complete subprogram descriptions for the DBUSER routines can be found in ISDOS Working Paper No. 88 (July, 1975). Also in this same working paper is a thorough explanation of the manner in which a user's program accesses the data base, sets currency indicators, etc.

7.2 DBLOW

The routines in DBLOW are low level routines used by DBUSER, which do much of the actual "work" involved in manipulating the data in the data base. They find control blocks, set pointers, compare items, find and release data base storage, and set the links in owner and member pointer areas. DBLOW also takes care of much of the actual page management facilities needed by DBUSER. DBLOW relies heavily upon DBTAB and DBRAND to access the data base. (DBRAND is used to get data base pages in and out of main memory, and DBTAB is used to directly access the data base control blocks and the object schema control blocks.)

7.2.1 DBLOW Subprogram Description Format

The DBLOW routines described here follow a standard format. This format entails the following:

1. Routine name
2. Calling convention
3. Purpose - a 1 line statement of the routine's function.
4. Description - a longer, narrative description of possible calling parameter values, algorithms used, special conditions, etc. This section may be omitted if the routine is very simple in function.

5. Arguments - A list of all the calling parameters are given. The format for each parameter is:
   -NAME (as found in the calling convention)
   -USAGE (Input, Output, Updated, Scratch)
   -TYPE (Integer, Real, Data Base Key, Character)
   -Description-(a short explanation of the way in which the argument is used.)

6. Errors - A listing of all possible errors is given.

7. Currency Indicators Used - a list of which currency indicators are referenced by the routine (Input) and which are reset by the routine (Output).

7.2.2 DBLOW Subprogram Descriptions

The following routines are located in DBLOW:

BLKFND	MPT
BLKREL	OPT
CITM	OWNLKR
DBERR	OWNLOK
DBKFND	RECLOK
DDLIN	REMOVE
FMKSDB	RPAGE
INSERT	SDBNXT
ITMLOK	SEARCH
MEMLKR	SETLOK
MEMLOK	WPAGE
MODPAG	ZMCHAN

CALLING CONVENTION: CALL BLKFND (RSIZE, PRHPTR, KEY)

PURPOSE: Find a block of storage in the data base.

DESCRIPTION: This routine checks the PHIMHS (maximum hole size) field of the current page. If the largest hole is smaller than the given blocksize, the remaining pages in core are searched. If a sufficiently large hole is not found, the entire data base is searched, beginning with the current page. If sufficient space is still not found, KEY is set to zero. If a block is found, the PRH pointer which indentifies the block is returned, along with the data base key of the block.

ARGUMENTS :

NAME	USAGE	TYPE	DESCRIPTION
RSIZE	INPUT	INTEGER	Size of the desired block (wor
PRHPTR	OUTPUT	INTEGER	PRH pointer of the allocated block.
KEY	OUTPUT	DBKEY	Key of the allocated block, or zero.

ERROR CONDITIONS :

CURRENCY INDICATORS USED :

CALLING CONVENTION: CALL BLKREL (KEY, IRC)

PURPOSE: Release a block of storage.

DESCRIPTION: The data base logical record identified by the given data base key is released, and becomes a hole. The hole is combined with other holes that preceed or succeed the new hole physically in the data base, and the combined hole is placed on the hole chain.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
KEY	INPUT	DBKEY	Key of record to be released.
IRC	OUTPUT	INTEGER	Error code

ERROR CONDITIONS:

0 - O.K.
1 - Invalid data base key given.

CURRENCY INDICATORS USED:

ADBMS DOCUMENTATION FORM

ROUTINE NAME: CITM

CALLING CONVENTION: CALL CITM (PRHPTR, IDBPTR, VALUE, ISW)

PURPOSE: Compare an item to a given value.

DESCRIPTION: The item identified by the given PRH and IDB pointer is compared to the given value for equality. The given IDB pointer is used to find the displacement of the item in the record identified by the given PRH pointer. VALUE may be integer, real, binary, or logical; it may be a data base key or a character array.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
PRHPTR	INPUT	INTEGER	PRH pointer to a physical record
IDBPTR	INPUT	INTEGER	IDB pointer to an item type
VALUE	INPUT	(see above)	Value to use in comparison
ISW	OUTPUT	INTEGER	Return Code: -1 ITEM less than VALUE 0 ITEM equal to VALUE +1 ITEM greater than VALUE

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

ADBMS DOCUMENTATION FORM

ROUTINE NAME: DBERR

CALLING CONVENTION: CALL DBERR (RTNNUM, IERR)

PURPOSE: Handle data base errors

DESCRIPTION: This routine is called from DBUSER routines, after an error is detected (see ISDOS Working Paper No. 88). The error number, routine name and routine nesting level is printed. See Appendix A for a listing of routine names and numbers.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RTNNUM	INPUT	INTEGER	Routine number which detected error (See appendix A)
IERR	INPUT	INTEGER	Error Code (see ISDOS Working Paper No. 88)

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

CALLING CONVENTION: PRHPTR = DBKFND (KEY)

PURPOSE: Find a PRH pointer for a given data base key.

DESCRIPTION: Integer Function which returns the PRH pointer associated with the logical record identified by the given data base key. If the key is invalid, a value of zero is returned. If the page on which the logical record is stored is not in core, it is read into core from the DBF. The current page is set to the page on which the logical record appears.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
KEY	INPUT	DBKEY	Data Base key
PRHPTR	OUTPUT	INTEGER	PRH pointer to a logical record, or zero.

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

CALLING CONVENTION: CALL DDLIN (LIONUM)

PURPOSE: Read in DDL tables (DBTF).

DESCRIPTION: The object schema control blocks and the DDL names vector are read into core using the given logical I/O unit number.

ARGUMENTS :

NAME	USAGE	TYPE	DESCRIPTION
LIONUM	INPUT	INTEGER	The logical I/O unit number to which the DBTF is attached

ERROR CONDITIONS :

CURRENCY INDICATORS USED :

ADBMS DOCUMENTATION FORM

ROUTINE NAME: FMKSDB

CALLING CONVENTION: CALL FMKSDB (SDBPTR, KEYCUR, KEYNXT, ISW, IRC)

PURPOSE: Find the next, previous, first, or last member of a set.

DESCRIPTION: The set identified by the given SDB pointer is searched for the next, previous, first, or last member (depending on KEYCUR and ISW). If found, the data base key of the located member is returned in KEYNXT. If not found, a value of zero is returned. The value of ISW indicates the direction (next or previous) in which the set is to be searched. The search is relative to the member identified by the given data base key (KEYCUR). If the value of KEYCUR is zero, the first or last member is sought.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
KEYCUR	INPUT	DBKEY	Key of member relative to which search is to be made; or zero for first or last.
KEYNXT	OUTPUT	DBKEY	Key of located member, or zero if not found.
ISW	INPUT	INTEGER	<0 find previous (last if KEYCUR = 0) >0 find next (first if KEYCUR = 0)
IRC	OUTPUT	INTEGER	Error Code

ERROR CONDITIONS:

-1 - Member not found (KEYNXT = 0)
0 - OK (KEYNXT = key of located member)
6 - Invalid member type (KEYCUR) for set type (SDBPTR)
7 - Invalid data base key (KEYCUR)
8 - No current owner of set type
12 - Record not member of set
99 - Catastrophe

CURRENCY INDICATORS USED: Current Owner of Set (INPUT)

CALLING CONVENTION: CALL INSERT (SDBPTR, KEYCUR, KEYNEW, ISW, IRC)

PURPOSE: Add a record to a set

DESCRIPTION: The record identified by KEYNEW is added as a member of the set identified by the given SDB pointer, relative to the record identified by KEYCUR. If the value of KEYCUR is zero, insertion is at the beginning or end of the set. The value of ISW indicates the direction of insertion: before KEYCUR or after KEYCUR (beginning or end if KEYCUR = 0).

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
KEYCUR	INPUT	INTEGER	Key of member relative to which insertion is made, or zero for beginning or end.
KEYNEW	INPUT	DBKEY	Key of record to be inserted.
ISW	INPUT	INTEGER	<0 insert before KEYCUR (beginning if KEYCUR = 0) >0 insert after KEYCUR (end if KEYCUR = 0)
IRC	OUTPUT	INTEGER	Error Code

ERROR CONDITIONS:

0 - OK
6 - Invalid member (KEYNEW) of given set.
7 - Invalid data base key (KEYNEW)
8 - No current owner of set
11 - Record already a member of set.
99 - Catastrophe

CURRENCY INDICATORS USED: Current Owner of Set (Input)

ADBMS DOCUMENTATION FORM

ROUTINE NAME: ITMLOK

CALLING CONVENTION: IDBPTR = ITMLOK (RDBPTR, NAMEL, NAME, NAMPTR)

PURPOSE: Find an item which belongs to a record type.

DESCRIPTION: Integer Function which returns the IDB pointer of the item which belongs to the record type identified by the given RDB pointer. The item is identified by the item name, which is stored in the character array NAME at index NAMPTR and length NAMEL. The name of each item in the record type is compared to the given name until the correct item is found, or until all the items are searched. If the item is not found, a value of zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type
NAMEL	INPUT	INTEGER	Length of the item name
NAME	INPUT	CHARACTER	Array where item DDL names are stored.
NAMPTR	INPUT	INTEGER	Index into NAME where the item name begins
IDBPTR	OUTPUT	INTEGER	IDB pointer to an item type, or zero if the item is not found.

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

CALLING CONVENTION: MDBPTR = MEMLKR (SDBPTR, RDBPTR)

PURPOSE: Look up an MDB of a set given an RDB pointer

DESCRIPTION: Integer Function which returns the MDB pointer for the record type which is a member of the set type identified by the given SDB pointer. The given RDB pointer is used to find the record name length and index pointer, then MEMLOK is referenced to find the MDB pointer. If the member is not found, a value of zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
RDBPTR	INPUT	INTEGER	RDB pointer to a record type
MDBPTR	OUTPUT	INTEGER	MDB pointer to a member control block.

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

ADBMS DOCUMENTATION FORM

ROUTINE NAME: MEMLOK

CALLING CONVENTION: MDBPTR = MEMLOK (SDBPTR, NAMEL, NAME, NAMPTR)

PURPOSE: Find an MDB of a given set

DESCRIPTION: Integer Function which returns the MDB pointer of the given member of the set identified by the given SDB pointer. The member is identified by its record name, which is stored in the character array NAME. The given member name is compared to the record name associated with each MDB in the set. If a pair of names match, the pointer to that MDB is returned; otherwise a zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type.
NAMEL	INPUT	INTEGER	Name length
NAME	INPUT	CHARACTER	Array where member DDL names are stored.
NAMPTR	INPUT	INTEGER	Index into NAME where the member record name begins
MDBPTR	OUTPUT	INTEGER	MDB pointer to a MDB or zero

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

CALLING CONVENTION: CALL MODPAG

PURPOSE: Set the page bits of the current page.

DESCRIPTION: This routine calls the DBTAB routine PHSBTS with a page bits value of '1'.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
none.			

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

ADBMS DOCUMENTATION FORM

ROUTINE NAME: MPT

CALLING CONVENTION: MEMPA = MPT (SDBPTR, KEY)

PURPOSE: Find the member pointer area for a member of a set.

DESCRIPTION: Integer Function which returns the pointer to the member pointers in the pointer area of the logical record identified by the given data base key. This record is assumed to be a legal member record of the set type identified by the given SDB pointer.

ARGUMENTS :

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
KEY	INPUT	DBKEY	Key for a member of the set
MEMPA	OUTPUT	INTEGER	Pointer to the member pointers in the pointer area of the member record.

ERROR CONDITIONS :

CURRENCY INDICATORS USED :

ADBMS DOCUMENTATION FORM

ROUTINE NAME: OPT

CALLING CONVENTION: OWNPA = OPT (SDBPTR, KEY)

PURPOSE: Find the owner pointer area for the owner of a set.

DESCRIPTION: Integer Function which returns the pointer to the owner pointers in the pointer area of the logical record identified by the given data base key. This record is assumed to be a legal owner record of the set type identified by the given SDB pointer.

ARGUMENTS :

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
KEY	INPUT	DBKEY	Key for an owner of the set
OWNPA	OUTPUT	INTEGER	Pointer to the owner pointers in the pointer area of the owner record.

ERROR CONDITIONS :

CURRENCY INDICATORS USED :

ADBMS DOCUMENTATION FORM

ROUTINE NAME: OWNLKR

CALLING CONVENTION: ODBPTR = OWNLKR (SDBPTR, RDBPTR)

PURPOSE: Look up an ODB of a set given an RDB pointer.

DESCRIPTION: Integer Function which returns the ODB pointer for the record type which is an owner of the set type identified by the given SDB pointer. The given RDB pointer is used to find the record name length and index pointer, then OWNLOK is referenced to find the ODB pointer. If the owner is not found, a value of zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
RDBPTR	INPUT	INTEGER	RDB pointer to a record type
ODBPTR	OUTPUT	INTEGER	ODB pointer to owner control block

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

CALLING CONVENTION: ODBPTR = OWNLOK (SDBPTR, NAMEL, NAME, NAMPTR)

PURPOSE: Find an ODB of a given set.

DESCRIPTION: Integer Function which returns the ODB pointer of the given owner of the set type identified by the given SDB pointer. The owner is identified by its record name, which is stored in the character array NAME. The given owner name is compared to the record name associated with each ODB in the set. If a pair of names match, the pointer to that ODB is returned; otherwise a value of zero is returned.

ARGUMENTS :

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
NAMEL	INPUT	INTEGER	Name length
NAME	INPUT	CHARACTER	Array where owner DDL names are stored.
NAMPTR	INPUT	INTEGER	Index into NAME where the owner record name begins.
ODBPTR	OUTPUT	INTEGER	ODB pointer to an ODB or zero.

ERROR CONDITIONS :

CURRENCY INDICATORS USED :

ADBMS DOCUMENTATION FORM

ROUTINE NAME: RECLOK

CALLING CONVENTION: RDBPTR = RECLOK (NAMEL, NAME, NAMPTR)

PURPOSE: Find a record type in the record table.

DESCRIPTION: Integer Function which returns the RDB pointer of the record type identified by the given record name length and index. The given record name is compared to the name of each record type in the data base. If the names match, the pointer to that record type is returned. If not found, a value of zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
NAMEL	INPUT	INTEGER	Name length
NAME	INPUT	CHARACTER	Array where DDL record names are stored
NAMPTR	INPUT	INTEGER	Index into NAME where the record name begins.
RDBPTR	OUTPUT	INTEGER	RDB pointer to a record type, or zero.

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

CALLING CONVENTION: CALL REMOVE (SDBPTR, KEYMEM, KEYNXT, IRC)

PURPOSE: Remove a single member of a set.

DESCRIPTION: The member record identified by KEYMEM is removed from the set identified by the given SDB pointer. First, the links in the member record pointer area are set to null. Then the member record is removed from the ownership of the owner of that member. The data base key of the next member of the set (KEYNXT) is returned. If KEYMEM was the last member of the set, a value of zero is returned in KEYNXT.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
KEYMEM	INPUT	DBKEY	Key of the member to be removed.
KEYNXT	OUTPUT	DBKEY	Key of the member which follows KEYMEM in the set, or zero if KEYMEM was last.
IRC	OUTPUT	INTEGER	Error Code

ERROR CONDITIONS:

0 - OK
12 - Record (KEYMEM) not a member of the set.
99 - Invalid links found.

CURRENCY INDICATORS USED:

ADBMS DOCUMENTATION FORM

ROUTINE NAME: RPAGE

CALLING CONVENTION: CALL RPAGE (PAGENO)

PURPOSE: Read a page into core.

DESCRIPTION: If the page with the given page number is not in core, it is read from the DBF into core, replacing, if necessary, another page already in core. The page with the given page number becomes the current page.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
PAGENO	INPUT	INTEGER	The page number of the desired data base page.

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

CALLING CONVENTION: SDBPTR = SDBNXT (SDBPTR)

PURPOSE: Find the next SDB in the set table.

DESCRIPTION: Integer Function which returns the SDB pointer of the set which follows the set identified by the given SDB pointer. If SDBPTR has an input value of zero, the pointer to the first SDB is returned. If SDBPTR points to the last set on input, a value of zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	UPDATED	INTEGER	INPUT: SDB pointer to a set (or zero for the first set) OUTPUT: SDB pointer to the next set (or zero if SDBPTR was the last.)

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

CALLING CONVENTION: CALL SEARCH(SDBPTR, VALUE, DUPSW, KEY, ISW, IRC)

PURPOSE: Find a member of a sorted set with a certain sort key value.

DESCRIPTION: The sorted set identified by the given SDB pointer is searched for a member with a sort key equal to the given sort key value (VALUE). The search begins with the current member of the set (or with the last member if the current member is null) and continues until the appropriate member is found, or until it is determined that there is no member of the set with the given sort key value.

If more than 1 member with the given sort key value is found, the search may continue until in order to find the first or last of the duplicates, depending on the value of DUPSW.

When the appropriate member is found, the data base key of the member is returned in KEY. This member becomes the current member of the set. If not found, the given sort key value goes either before or after the member identified by KEY, depending on the value of ISW. IRC is set to -1.

VALUE may only be of type integer or character array.

It is assumed that the set is sorted and that it has a current owner. The search is linear.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer a set type (sorted
VALUE	INPUT	(see above)	Sort key value to look for.
DUPSW	INPUT	INTEGER	What to do with duplicates -1 return key of first duplicate 0 don't care +1 return key of last duplicate.
KEY	OUTPUT	DBKEY	Data base key of appropriate member (see ISW)
ISW	OUTPUT	INTEGER	Relative location of KEY to VALUE -1 VALUE not found, VALUE goes before KEY (beginnin if KEY = 0) 0 VALUE found, KEY is membe (see DUPSW) +1 VALUE not found, VALUE goes after KEY (end if KEY = 0)
IRC	OUTPUT	INTEGER	Error Code

ERRORS:

-1 - Value not found (see ISW).
0 - OK
8 - No current owner of set
99 - Set out of sequence

CURRENCY INDICATORS USED:

Current Owner of Set (Input)
Current Member of Set (Output)

ADBMS DOCUMENTATION FORM

ROUTINE NAME: SETLOK

CALLING CONVENTION: SDBPTR = SETLOK (NAMEL, NAME, NAMPTR)

PURPOSE: Find a set type in the set table.

DESCRIPTION: Integer Function which returns the SDB pointer of the set type identified by the given set name length and index. The given set name is compared to the name of each set in the data base. If the names match, the pointer to that set type is returned. If not found, a value of zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
NAMEL	INPUT	INTEGER	Name length
NAME	INPUT	CHARACTER	Array where DDL set names are stored.
NAMPTR	INPUT	INTEGER	Index into NAME where the set name begins.
SDBPTR	OUTPUT	INTEGER	SDB pointer to a set type, or zero.

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

ROUTINE NAME: WPAGE

CALLING CONVENTION: CALL WPAGE

PURPOSE: Write the current page (if it has been modified)

DESCRIPTION: The page bits of the current page are checked. If non-zero, the page is rewritten into the DBF.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
none.			

ERROR CONDITIONS:

CURRENCY INDICATORS USED:

CALLING CONVENTION: CALL ZMCHAN (SDBPTR, KEY, IRC)

PURPOSE: Remove all the members from the owner of a set.

DESCRIPTION: Each member of the set identified by the given SDB pointer, and which has as its owner the record identified by the given data base key, is removed from the set.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set
KEY	INPUT	DBKEY	Key of an owner of the set
IRC	OUTPUT	INTEGER	Error code

ERROR CONDITIONS:

00 - OK
99 - Invalid links found in member pointers.

CURRENCY INDICATORS USED:

7.3 DBTAB

The DBTAB routines are used to reference and update the seven types of control blocks described in sections 3 and 4. They can be divided into two groups: routines that reference particular fields in the control blocks, and routines that change particular fields.

The routines that only reference the data base control blocks are all FORTRAN integer functions. The first three letters of the routine name identify the control block they use, and the last three letters identify the particular field. For example, PHIBTS is an integer function which returns the page bits field of a given PHI. Note that the function name is the same as the name of the field that it references.

All the routines that change fields in the control blocks are FORTRAN subroutines. Here again, the subroutine name is the same as the name of the field it changes, EXCEPT that the third letter of the name is an 'S', i.e., the subroutine PHSBTS sets the page bits for a given PHI.

7.3.1 DBTAB Subprogram Descriptions Format

The DBTAB subprogram description format is the same as that described in section 7.2.1.

7.3.2 DBTAB Subprogram Descriptions

The following routines are available in DBTAB:

Control Block	Function	Subroutine
IDB:	IDBTYP	
	IDBBTS	
	IDBMLI	
	IDBMVD	
	IDBDDP	
	IDBDSP	
	IDBMTV	
MDB:	MDBDSP	
ODB:	ODBDSP	
PHI:	PHIPNM	
	PHIMHS	PHSMHS
	PHIHCH	PHSHCH
	PHIBTS	PHSBTS
PRH:	PRHLEN	PRSLEN
	PRHLNH	PRSLNH
	PRHRDB	PRSRDB

Control Block	Function	Subroutine
RDB:	RDBNMP	
	RDBNML	
	RDBCUR	RDSCUR
	RDBITM	
	RDBMLR	
	RDBDAL	
	RDBOWN	
	RDBMEM	
	RDBBTS	
	RDBDIS	
SDB:	SDBCRO	SDSCRO
	SDBCRM	SDSCRM
	SDBSNP	
	SDBSNL	
	SDBSKM	
	SDBOBT	
	SDBSKT	

ADBMS DOCUMENTATION FORM

ROUTINE NAME: IDBTYP

CALLING CONVENTION: I = IDBTYP(IDBPTR)

PURPOSE: Integer Function which returns the item type from the IDB identified by the given IDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
IDBPTR	INPUT	INTEGER	IDB pointer to an item-type
I	OUTPUT	INTEGER	Item Type: 1. or 2. Integer
			4 or 8 Real
			16 Binary
			32 Data Base Key
			64 Logical
			128 Character

CALLING CONVENTION: I = IDBBTS (IDBPTR)

PURPOSE: Integer Function which returns the item bits from the IDB identified by the given IDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
IDBPTR	INPUT	INTEGER	IDB pointer to an item type.
I	OUTPUT	INTEGER	Item Bits: 1 Single item
			2 Fixed item
			4 Variable item
			8 Depended on ite

CALLING CONVENTION: I = IDBMLI (IDBPTR)

PURPOSE: Integer Function which returns the value of the maximum length of the item from the IDB identified by the given IDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
IDBPTR	INPUT	INTEGER	IDB pointer to an item-type
I	OUTPUT	INTEGER	Maximum length of the item (words)

CALLING CONVENTION: I = IDBMVD (IDBPTR)

PURPOSE: Integer Function which returns for a repeating item the maximum value of the item's depending item. This value is taken from the IDB identified by the given IDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
IDBPTR	INPUT	INTEGER	IDB pointer to an item type.
I	OUTPUT	INTEGER	Maximum value of depending ite

CALLING CONVENTION: I = IDBDDP (IDBPTR)

PURPOSE: Integer Function which returns for a repeating item the displacement (into the data area) of the item's depending item. This displacement is taken form the IDB identified by the given IDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
IDBPTR	INPUT	INTEGER	IDB pointer to an item-type
I	OUTPUT	INTEGER	Displacement of the depending item (words).

CALLING CONVENTION: I = IDBDSP (IDBPTR)

PURPOSE: Integer Function which returns the item displacement (into the data area). This displacement is taken from the IDB identified by the given IDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
IDBPTR	INPUT	INTEGER	IDB pointer to an item-type
I	OUTPUT	INTEGER	Displacement of the item (wor

CALLING CONVENTION: I = IDBTMV (IDBPTR)

PURPOSE: Integer Function which returns the value of the item type modifier from the IDB identified by the given IDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
IDBPTR	INPUT	INTEGER	IDB pointer to an item-type
I	OUTPUT	INTEGER	Item type modifier.

CALLING CONVENTION: I = MDBDSP(MDBPTR)

PURPOSE: Integer Function which returns the displacement (into the pointer area) of the member pointers from the MDB identified by the given MDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
MDBPTR	INPUT	INTEGER	MDB pointer to a member control block
I	OUTPUT	INTEGER	Displacement of the member pointers (words)

CALLING CONVENTION: I = ODBDSP (ODBPTR)

PURPOSE: Integer Function which returns the displacement (into the pointer area) of the owner pointers from the ODB identified by the given ODB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
ODBPTR	INPUT	INTEGER	ODB pointer to an owner control block
I	OUTPUT	INTEGER	Displacement of the owner pointers (words)

CALLING CONVENTION: I = PHIPNM (ARG)

PURPOSE: Integer Function which returns the page number of the current page.

Note that ARG should always contain the value '1'.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
ARG	INPUT	INTEGER	A dummy argument which has the value '1'.
I	OUTPUT	INTEGER	Page number of the current page.

CALLING CONVENTION: I = PHIMHS (ARG)

PURPOSE: Integer Function which returns the value for the maximum hole size of the current page.

Note that ARG should always contain the value '1'.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
ARG	INPUT	INTEGER	Dummy argument - value always '1'.
I	OUTPUT	INTEGER	Maximum hole size of the current page.

CALLING CONVENTION: I = PHIHCH (ARG)

PURPOSE: Integer Function which returns, for the current page, the displacement on that page of the first hole on the hole chain. If there are no holes, a value of zero is returned.

Note that ARG should always contain the value '1'.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
ARG	INPUT	INTEGER	Dummy argument - value always '1'.
I	OUTPUT	INTEGER	Displacement of first hole on the current page (words), or zero.

CALLING CONVENTION: I = PHIBTS(ARG)

PURPOSE: Integer Function which returns the value of the page bits for the current page.

Note that ARG should always contain a value of '1'.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
ARG	INPUT	INTEGER	Dummy Argument - value always '1'.
I	OUTPUT	INTEGER	Page bits of current page.

CALLING CONVENTION: CALL PHSMHS(ARG, MHS)

PURPOSE: Set the value of the maximum hole size of the current page to the given hole size. If there are no holes, the maximum hole size should be set to zero.

Note that ARG should always contain a value of '1'.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
ARG	INPUT	INTEGER	Dummy Argument - value always '1'.
MHS	INPUT	INTEGER	Maximum hole size or zero.

CALLING CONVENTION: CALL PHSHCH(ARG, HCH)

PURPOSE: Set the PHIHCH field (displacement of the first hole on the hole chain) of the current page to the given displacement. If there is no hole, the PHIHCH field should be set to zero.

Note that ARG should always contain a value of '1'.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
ARG	INPUT	INTEGER	Dummy Argument - value always '1'.
HCH	INPUT	INTEGER	Displacement of the first hole (words), or zero.

CALLING CONVENTION: CALL PHSBTS(ARG, BTS)

PURPOSE: Set the PHIBTS field (page bits) of the current page to the given page bits value.

Note that ARG should always contain the value '1'.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
ARG	INPUT	INTEGER	Dummy Argument - value is '1'.
I	INPUT	INTEGER	Page bits

CALLING CONVENTION: I = PRHLEN(PRHPTR)

PURPOSE: Integer Function which returns the logical record length from the PRH identified by the given PRH pointer. The length of the record may include up to 2 extra words which follow the record as padding (see section 5.2).

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
PRHPTR	INPUT	INTEGER	PRH pointer to a logical record
I	OUTPUT	INTEGER	Logical record length (words)

CALLING CONVENTION: PRHPT2 = PRHLNH(PRHPT1)

PURPOSE: Integer Function which returns the pointer to the hole on the hole chain which follows the current hole. The given PRH pointer (PRHPT1) points to the current hole on the current page. If the current hole is the last hole on the chain, the value returned is zero.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
PRHPT1	INPUT	INTEGER	PRH pointer to the current hole.
PRHPT2	OUTPUT	INTEGER	PRH pointer to the next hole, or zero.

CALLING CONVENTION: RDBPTR = PRHRDB(PRHPTR)

PURPOSE: Integer Function which returns the RDB pointer from the PRH identified by the given PRH pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
PRHPTR	INPUT	INTEGER	PRH pointer to a logical record
RDBPTR	OUTPUT	INTEGER	RDB pointer to a record type

CALLING CONVENTION: CALL PRSLEN(PRHPTR, LEN)

PURPOSE: Set the PRHLEN field (the logical record length) in the PRH identified by the given PRH pointer to the given record length.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
PRHPTR	INPUT	INTEGER	PRH pointer to a logical record
LEN	INPUT	INTEGER	Length of the record (words)

ROUTINE NAME: PRSLNH

CALLING CONVENTION: CALL PRSLNH(PRHPTR, NHPTR)

PURPOSE: Set the PRHLNH field (link to the next hole) in the PRH identified by the given PRH pointer to the given data base hole pointer, which points to the PRH of the hole. If the current hole is the last hole on the chain, the link to the next hole should be set to zero.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
PRHPTR	INPUT	INTEGER	PRH pointer to a data base hole
NHPTR	INPUT	INTEGER	PRH pointer to the next hole on the hole chain, or zero.

ROUTINE NAME: PRSRDB

CALLING CONVENTION: CALL PRSRDB(PRHPTR, RDBPTR)

PURPOSE: Set the PRHRDB field (RDB pointer) in the PRH identified by the given PRH pointer to the given RDB pointer. This RDB pointer should point to the record type of which the logical record, identified by the given PRH pointer, is an occurrence.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
PRHPTR	INPUT	INTEGER	PRH pointer to a logical record
RDBPTR	INPUT	INTEGER	RDB pointer to a record type

CALLING CONVENTION: I = RDBNMP(RDBPTR)

PURPOSE: Integer Function which returns the record name index pointer from the RDB identified by the given RDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type
I	OUTPUT	INTEGER	Record name index pointer into NAMES.

ROUTINE NAME: RDBNML

CALLING CONVENTION: I = RDBNML(RDBPTR)

PURPOSE: Integer Function which returns the record name length from the RDB identified by the given RDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type
I	OUTPUT	INTEGER	Record name length (words)

CALLING CONVENTION: KEY = RDBCUR(RDBPTR)

PURPOSE: Integer Function which returns the value of the data base key of the current record of the record type identified by the given RDB pointer. If there is no current record, a value of zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type
KEY	OUTPUT	DBKEY	Data base key of the current record, or zero.

CALLING CONVENTION: I = RDBITM(RDBPTR)

PURPOSE: Integer Function which returns the value of the number of items in the record type identified by the given RDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type.
I	OUTPUT	INTEGER	Number of items in the record type.

CALLING CONVENTION: I = RDBMLR(RDBPTR)

PURPOSE: Integer Function which returns the value of the maximum length of the data base record from the RDB identified by the given RDB pointer. (See Section 5.2.)

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type
I	OUTPUT	INTEGER	Maximum length of the record type (words).

ADBMS DOCUMENTATION FORM

ROUTINE NAME: RDBDAL

CALLING CONVENTION: I = RDBDAL(RDBPTR)

PURPOSE: Integer Function which returns the value of the logical record data area length from the RDB identified by the given RDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type.
I	OUTPUT	INTEGER	Data area length (words).

CALLING CONVENTION: I = RDBOWN(RDBPTR)

PURPOSE: Integer Function which returns the value for the number of sets of which the record type identified by the given RDB pointer is a legal owner.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type.
I	OUTPUT	INTEGER	Number of sets of which record type is an owner.

CALLING CONVENTION: I = RDBMEM(RDBPTR)

PURPOSE: Integer Function which returns the value for the number of sets of which the record type identified by the given RDB pointer is a legal member.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type
I	OUTPUT	INTEGER	Number of sets of which record type is a member.

ROUTINE NAME: RDBBTS

CALLING CONVENTION: I = RDBBTS(RDBPTR)

PURPOSE: Integer Function which returns the value of the record bits from the RDB identified by the given RDB pointer.

ARGUMENTS:	NAME	USAGE	TYPE	DESCRIPTION
	RDBPTR	INPUT	INTEGER	RDB pointer to a record type
	I	OUTPUT	INTEGER	Record bits: 1 - fixed length 2 - variable length 4 - system record

CALLING CONVENTION: I = RDBDIS(RDBPTR)

PURPOSE: Integer Function which returns the value of the displacement (into the logical record) of the data area for the record type identified by the given RDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type.
I	OUTPUT	INTEGER	Displacement of the data area (words).

ADBMS DOCUMENTATION FORM

ROUTINE NAME: RDSCUR

CALLING CONVENTION: CALL RDSCUR(RDBPTR, KEY)

PURPOSE: Set the current record indicator of the record type identified by the given RDB pointer to the given data base key value. The current record indicator is set to null by calling this routine with a data base key value of zero.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
RDBPTR	INPUT	INTEGER	RDB pointer to a record type
KEY	INPUT	DBKEY	Data base key of the current record, or zero.

ROUTINE NAME: SDBCRO

CALLING CONVENTION: KEY = SDBCRO(SDBPTR)

PURPOSE: Integer Function which returns the value of the data base key of the current owner of the set type identified by the given SDB pointer. If the current owner indicator of this set type is null, a value of zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type.
KEY	OUTPUT	DBKEY	Key of the current owner, or zero.

CALLING CONVENTION: KEY = SDBCRM(SDBPTR)

PURPOSE: Integer Function which returns the value of the data base key of the current member of the set type identified by the given SDB pointer. If there is no current member, a value of zero is returned.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
KEY	OUTPUT	DBKEY	Key of the current member, or zero.

CALLING CONVENTION: I = SDBSNP(SDBPTR)

PURPOSE: Integer Function which returns the sort key name index pointer from the SDB identified by the given SDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
I	OUTPUT	INTEGER	Sort key name index pointer into NAMES.

CALLING CONVENTION: I = SDBSNL(SDBPTR)

PURPOSE: Integer Function which returns the sort key name length from the SDB identified by the given SDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
I	OUTPUT	INTEGER	Sort key name length

ROUTINE NAME: SDBSKM

CALLING CONVENTION: I = SDBSKM(SDBPTR)

PURPOSE: Integer Function which returns the value of the sort key type modifier from the SDB identified by the given SDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
I	OUTPUT	INTEGER	Sort key type modifier

CALLING CONVENTION: I = SDBOBT(SDBPTR)

PURPOSE: Integer Function which returns the value of the order bits from the SDB identified by the given SDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
I	OUTPUT	INTEGER	Order bits: 1 FIFO
			2 LIFO
			4 NEXT
			8 PRIOR
			16 IMMATERIAL
			32 SORTED ON ...

CALLING CONVENTION: I = SDBSKT(SDBPTR)

PURPOSE: Integer Function which returns the sort key type from the SDB identified by the given SDB pointer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
I	OUTPUT	INTEGER	Sort key type (see IDBTYP)

CALLING CONVENTION: CALL SDSCRO(SDBPTR, KEY)

PURPOSE: Set the current owner indicator of the set type identified by the given SDB pointer to the given data base key value. The current owner indicator is set to null by calling this routine with a data base key value of zero.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
KEY	INPUT	DBKEY	Key of the current owner, or zero.

CALLING CONVENTION: **CALL SDSCRM(SDBPTR, KEY)**

PURPOSE: Set the current member indicator of the set type identified by the given SDB pointer to the given data base key value. The current member indicator is set to null by calling this routine with a data base key value of zero.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
SDBPTR	INPUT	INTEGER	SDB pointer to a set type
KEY	INPUT	DBKEY	Key of current member, or zero.

7.4 DBRAND

There are 5 subroutines located in DBRAND, whose function is to provide random input/output between main memory and the DBF.

A random I/O file must be opened before it can be accessed by the user program; this is done with RANDOP. To insure that the data base has been fully updated, the file must also be closed before the user program ends, using RANDCL. If this is not done, the data base can easily become inconsistent.

The other three routines, RANDRD, RANDRW, and RANDWT, provide facilities to read, rewrite, and write specific pages of the data base.

7.4.1 DBRAND Subroutine Description Format

See section 7.2.1

7.4.2 DBRAND Subroutine Descriptions

The following routines are available in DBRAND:

RANDCL
RANDOP
RANDRD
RANDRW
RANDWT

CALLING CONVENTION: CALL RANDCL(FDESG)

PURPOSE: To close a file opened for random I/O.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
FDESG	INPUT	INTEGER	File designator returned by RANDOP

CALLING CONVENTION: CALL RANDOP(LIONUM, FDESG)

PURPOSE: To open a file for random I/O.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
LIONUM	INPUT	INTEGER	Logical I/O unit number to use for the DBF.
FDESG	OUTPUT	INTEGER	File designator used by other DBRAND routines.

CALLING CONVENTION: CALL RANDRD(FDESG, PAGENO, BUFFER)

PURPOSE: Read a given page from a random I/O file into a given buffer.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
FDESG	INPUT	INTEGER	File designator returned by RANDOP.
PAGENO	INPUT	INTEGER	Page number to read.
BUFFER	OUTPUT	INTEGER	Array dimensioned to the page size (in words), into which the page is read.

CALLING CONVENTION: CALL RANDRW(FDESG, PAGENO, BUFFER)

PURPOSE: Rewrite a given page into a random I/O file from a buffer. This page is assumed to exist in the DBF.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
FDESG	INPUT	INTEGER	File designator returned by by RANDOP
PAGENO	INPUT	INTEGER	Page number to be rewritten
BUFFER	INPUT	INTEGER	Array dimensioned to the page size (in words), from which the page is rewritten.

CALLING CONVENTION: CALL RANDWT(FDESG, PAGENO, BUFFER)

PURPOSE: Write a given page into a random I/O file from a buffer. This page is a new page being added to the DBF.

ARGUMENTS:

NAME	USAGE	TYPE	DESCRIPTION
FDESG	INPUT	INTEGER	File designator returned by RANDOP.
PAGENO	INPUT	INTEGER	Page number to be written
BUFFER	INPUT	INTEGER	Array dimensioned to the page size (in words) from which the page is written.

7.5 Block Data for the DBCS

A block data must be supplied for the DBCS, to initialize certain constants and user supplied values. Figure 7.5 is an example block data for an IBM System 360/370 installation. The information in common areas BLENS and DBSWS are dependent on the data base system, and should not be changed. The information in areas MACHIN and PAGINF is installation dependent, and should be adjusted accordingly.

7.5.1 MACHIN

The number of characters per word for the particular machine being used is supplied in the variable NCW, located in named common MACHIN.

7.5.2 BLENS

Common BLENS contains the lengths of the control blocks in the data base. The block sizes remain invariant throughout the data base.

7.5.3 PAGINF

Certain information must be supplied by for the DBCS page management system. This includes values for NPAGES, PAGSIZ, MPICOR, and an initial value for CURPNO. The array PREF should be dimensioned here, to the value of MPICOR.

7.5.4 PAGE

The common area PAGE, although not initialized, is included here to show the size of the page buffer. PAGE should be dimensioned to a value of PAGSIZ*MPICOR.

7.5.5 DBSWS

The information found in common DBSWS is used by the routines in DBUSER (see section 7.1.1 of this working paper). OPENSW is a logical switch which indicates whether the data base has been opened or not. LEVEL is a counter used by the DBCS to return error information to the user.

7.6 DBCS Dependence upon routine library SLIB

The routines in DBUSER, DBLOW, and DBTAB are all written in machine independent FORTRAN. To facilitate this independence, DBCS utilizes a machine dependent routine library, SLIB. The SLIB routines for an installation are all written in the machine language of that particular computer; they are documented in ISDOS Working Paper No. 111, May 1975.

The following is a list of routines located in SLIB.

SUBROUTINES	INTEGER FUNCTIONS
CPUSEC	IAND
GETDAT	IBITS
GETTIM	IBYTE
ITOC	IBYTE1
LEFTI	ICNC1
RIGHTI	ICSC1
SBITS	IHASH
SMOVE	IOR
	ISCOMP
	ISHFTL
	ISHFTR
	LEFT
	RIGHT

```
#slist -blkdata
>      1              BLOCK DATA
>      2      C
>      3              COMMON/MACHIN/NCW
>      4              INTEGER NCW
>      5      C
>      6              COMMON/BLENS /RDBLEN,IDBLEN,SDBLEN,ODBLEN,MDBLEN,
>      7             & PHILEN,PRHLEN
>      8              INTEGER RDBLEN,IDBLEN,SDBLEN,ODBLEN,MDBLEN,
>      9             & PHILEN,PRHLEN
>     10      C
>     11              COMMON/PAGINF/NPAGES,PAGSIZ,FILE,CURPNO,MPICOR,NPICOR,NCUR
>     12             & PHICUR,PREF(16)
>     13              INTEGER NPAGES,PAGSIZ,FILE,CURPNO,MPICOR,NPICOR,NCUR,
>     14             & PHICUR,PREF
>     15      C
>     16              COMMON/PAGE  /PAGE(16384)
>     17              INTEGER PAGE
>     18      C
>     19              COMMON/DBSWS /OPENSW,LEVEL
>     20              INTEGER LEVEL
>     21              LOGICAL OPENSW
>     22      C
>     23      C.....................................................................
>     24      C
>     25              DATA NCW/4/
>     26      C
>     27              DATA RDBLEN/6/
>     28              DATA IDBLEN/5/
>     29              DATA SDBLEN/7/
>     30              DATA ODBLEN/2/
>     31              DATA MDBLEN/2/
>     32              DATA PHILEN/2/
>     33              DATA PRHLEN/1/
>     34      C
>     35              DATA NPAGES/2/
>     36              DATA PAGSIZ/1024/
>     37              DATA MPICOR/16/
>     38              DATA CURPNO/0/
>     39      C
>     40              DATA OPENSW/.FALSE./
>     41              DATA LEVEL/0/
>     42      C
>     43              END
#END OF FILE
#
```

FIGURE 7.5

8. DATA BASE UTILITY PROGRAMS

ADBMS uses three utility programs to generate, initialize, and report on data bases. All three programs are written totally in FORTRAN. The remainder of this section describes each program in detail.

8.1 DDLA (Data Description Language Analyzer)

This program accepts a data base description (DDL) as input and generates three types of output: an analysis of the data base, in the form of a printed summary of the DDL tables; a FORTRAN BLOCK DATA source subprogram which is to be used in conjunction with the DBCS in a user's program; and the DDL tables for the data base (DBTF).

The program can be divided into four logical sections:

- Input of the DDL and generation of the DDL tables
- Summary Report for the DDL tables
- Generation and output of the BLOCK DATA
- Output of the DDL tables into the DBTF

8.1.1 Input of the DDL and generation of the DDL tables

DDLA begins by initializing several variables, and by setting up the RDB for the SYSTEM record. It then enters a loop which consists of:

- Reading an input line (form the DDL file)
- Echoing the input line
- Determining what type DDL card the input line is (see ISDOS Working Paper No. 88)
- Setting up the DDL tables for that card type

RECTAB, SETTAB, and NAMES are all set up in this way. The information from the DDL is stored in the tables in the appropriate fields. Once the entire DDL has been read in and stored, DDLA goes back over RECTAB and SETTAB to compute any remaining data that needs to be supplied. DDLA computes for each RDB the maximum record length of the record type, and the displacement of the data area. DDLA also computes the displacement (into the physical record) of each item in the record type, and of the depending item, if one exists.

For each SDB in the set table, DDLA sets up the sort key (if the set is sorted), and if SYSTEM is a legal owner of the set, the current owner is set to the SYSTEM record.

8.1.2 Summary Report

The next step that DDLA performs is to print a summary report of RECTAB and SETTAB. The report consists of one section for every record type, and one section for every set type. The following table shows which

information is printed for each record type. (The name of the control block data field is indicated here, when appropriate. (See section 3.1 of this working paper).

For each RDB:

RECORD NAME (RDBNMP, RDBNML)
POINTER INTO RECTAB OF RDB
NUMBER OF ITEMS (RDBMLR)
MAX LENGTH (RDBMLR)
POINTER AREA LENGTH (RDBPAL)
DATA AREA LENGTH (RDBDAL)
DATA DISPLACEMENT (RDBDIS)
RECORD BITS (RDBBTS)

NUMBER OF SETS OF WHICH THIS RECORD TYPE IS A LEGAL OWNER (RDBOWN)
For each set of which this record is a legal owner:
SET NAME (SDBNMP, SDBNML)
DISP OF OWNER POINTERS(ODBDSP)
SET ORDERING(SDBOBT)
SORT KEY(SDBSNP, SDBSNL)
POINTER INTO SETTAB OF ODB

NUMBER OF SETS OF WHICH THIS RECORD TYPE IS A LEGAL MEMBER(RDBMEM)

For each set of which this record is a legal member:

SET NAME (SDBNMP, SDBNML)
DISP OF MEMBER POINTERS (MDBDSP)
SET ORDERING (SDBOBT)
SORT KEY (SDBSNP, SDBSNL)
POINTER INTO SETTAB OF MDB

For each item that belongs to this record type:

ITEM NAME (IDBNMP, IDBNML)
ITEM TYPE (IDBTYP)
ITEM MODIFIER VALUE (IDBTMV)
ITEM BITS (IDBBTS)
POINTER INTO RECTAB OF IDB
MAX LENGTH OF ITEM (IDBMLI)
DISP OF ITEM (IDBDSP)
MAX VALUE OF DEPENDING ITEM (IDBMVD)
DISP OF DEPENDING ITEM (IDBDDP)

The next table shows which information is printed for each set type. (Again, the control block data field name is given here, when appropriate.)

For each SDB:

SET NAME (SDBNMP, SDBNML)
POINTER INTO SETTAB OF SDB
SET ORDERING (SDBOBT)
SORT KEY NAME (SDBSNP, SDBSNL)
SORT KEY LENGTH (SDBSKL)

NUMBER OF POSSIBLE OWNER RECORD TYPES (SDBNMO)

For each possible owner record type:
OWNER NAME (RDBNMP, RDBNML)
DISP OF OWNER POINTERS (ODBDSP)
POINTER INTO SETTAB OF ODB

NUMBER OF POSSIBLE MEMBER RECORD TYPES (SDBNMM)
For each possible member record type:
MEMBER NAME (RDBNMP, RDBNML)
DISP OF MEMBER POINTERS (MDBDSP)
POINTER INTO SETTAB OF MDB

8.1.3 Generation and output of BLOCK DATA

DDLA next generates a FORTRAN BLOCK DATA subprogram, which defines the storage area needed for the data base tables. The FORTRAN source code is generated for the common areas RECTAB, SETTAB, and NAMES, and the length of the corresponding vectors are initialized in the variables RECLEN, SETLEN, and NAMLEN. Note that values for RECLEN and SETLEN are in words, but the value of NAMLEN is in bytes.

The BLOCK DATA may be punched onto cards, or can be stored in a file. The statements are syntactically correct and complete, and can be compiled without alteration.

8.1.4 Output of the data base tables

The last procedure DDLA performs is to write out the data base tables, which it has generated, into the DBTF. See section 3.4 of this working paper for a description of the DBTF structure.

8.1.5 Logical input/output unit numbers for DDLA

Currently, the following FORTRAN logical I/O unit numbers are used for the indicated purposes:

5 - Input of the DDL
6 - Printed Output. This includes the echo of the DDL input, and the DDL analysis summary report.
7 - FORTRAN BLOCK DATA
8 - DBTF (the DDL tables)

8.2 DBIN (Data Base Initializer)

DBIN uses the DBTF generated by DDLA to initialize a data base. The size of the initialized data base is determined by the NPAGES card in the DDL.

Each page is first completely filled with zeroes. DBIN sets up the PHI on each page with the correct page and hole information. In addition, one occurrence of the SYSTEM record is placed at the beginning of the first page. Each page is set up and then written out into the DBF using the DBRAND routine RANDWT.

The following FORTRAN logical I/O unit numbers are used by DBIN:

2 - Output of the DBF
3 - Input of the DBTF

8.3 DBSM (Data Base Summary)

DBSM is a program which is executed using a populated data base. It produces three types of summaries:

- Page Summary
- Page Summary Statistics
- Record Summary

8.3.1 Page Summary

DBSM first prints out a summary for each page, giving the following information:

PAGE NUMBER
MAXIMUM HOLE SIZE
NUMBER OF HOLES
TOTAL SPACE USED BY HOLES
PERCENTAGE SPACE USED BY HOLES
NUMBER OF RECORDS
TOTAL SPACE USED BY RECORDS
PERCENTAGE SPACE USED BY RECORDS

8.3.2 Page Summary Statistics

Following the page summary, DBSM prints out a similar set of information for the data base as a whole. This includes the following:

PAGE SIZE
NUMBER OF PAGES

TOTAL NUMBER OF HOLES
TOTAL HOLE SIZE
TOTAL PERCENTAGE OF HOLES

TOTAL NUMBER OF RECORDS
TOTAL RECORD SIZE
TOTAL PERCENTAGE OF RECORDS

8.3.3 Record Summary

The third set of summary information produced by DBSM is a summary of space used by each record type in the data base. The following information is given for each record type:

RECORD NAME
SIZE OF POINTER AREA
SIZE OF DATA AREA
TOTAL SIZE OF PHYSICAL RECORD
NUMBER OF OCCURRENCES
TOTAL SPACE USED
PERCENTAGE SPACE USED

8.3.4 Logical Input/Output Unit Numbers Used by DBSM

The following FORTRAN logical I/O unit numbers are used by DBSM:

2 - Input of DBF
3 - Input of DBTF
6 - Output of printed summaries

GLOSSARY

ADBMS - The data base management system described by this working paper

Control block - An area of storage in the DBF or DBTF whose fields contain pointers or control information for the data base management system.

CURPNO - A FORTRAN variable, whose value is the data base page number of the current page.

Current member of a set - A conceptual pointer to one particular member of a set. Each set in the data base has a current member, which may be set to null.

Current owner of a set - A conceptual pointer to one particular owner of a set. Each set in the data base has a current owner, which may be be set to null.

Current page - A conceptual pointer to one particular page in the data base which is in main memory.

Current record of a record type - A conceptual pointer to one particular record occurrence of a record type. Each record type in the data base has a current record, which may be set to null.

Data base data record - See logical record.

Data base hole - A block of storage in the DBF which is available to the DBCS for allocation as a logical record.

Data base key - A fullword of data used to uniquely identify a logical record.

Data base page - A block of storage that the data base is divided into. It is also the amount of the DBF moved in and out of main memory at one time.

DBCS - The Data Base Control System; a collection of FORTRAN routines which interface with a user's program to access a data base.

DBF - The Data Base File; the name of the place where the data and data base control blocks are stored.

DBIN - A FORTRAN program which uses an object schema to initialize a data base.

DBLOW - The name of a set of FORTRAN routines that are part of the DBCS. They are lower level routines used to access the data base.

DBRAND - The name of a set of FORTRAN routines which perform random input/output operations. These routines are part of the DBCS.

DBSM - A FORTRAN program which prints summary information on the utilization of a data base.

DBTAB - The name of a set of FORTRAN routines which access the data base control blocks. These routines are part of the DBCS.

DBTF - The Data Base Table File; the name of the place where the data base tables are stored.

DBUSER - The name of a set of FORTRAN routines which interface directly with the user's program. These routines are part of the DBCS.

DDL - The Data Description Language; a formal language used to describe a data base in source form.

DDLA - A FORTRAN program which uses a data base description (DDL) to generate the data base tables (object schema).

Hole - See data base hole.

Hole chain - A linked list of all the data base holes on a data base page arranged in increasing order of size.

IDB - The Item Description Block; an object schema control block used to logically describe an item.

Item - The elementary data unit, from which all other types of structures are ultimately composed. (An item is sometimes called a field, data item, or element in other DBMS.)

Key - See data base key.

Logical record - A block of storage in the DBF in which data is stored. A logical record consists of a PRH, a pointer area, and a data area.

MDB - The Member Description Block; an object schema control block used to identify members of a set.

MPICOR - A FORTRAN variable, whose value is the maximum number of data base pages allowed in main memory at one time.

NAMES - A character vector which contains the names of all sets, records, and items described in a DDL

NCUR - A FORTRAN variable, whose value specifies the sequence number in main memory of the current page.

NDBKF - A FORTRAN variable, whose value is the number of times the current page has been set as a result of calling DBKFND.

NPICOR - A FORTRAN variable, whose value is the current number of data base pages in main memory.

Object schema - The logical description of the data base.

ODB - The Owner Description Block; an object schema control block used to identify owners of a set.

Page - See data base page.

PAGE - A FORTRAN vector, which is used as the data base buffer in main memory.

PHI - The Page Header Information; a data base control block which appears at the beginning of each data base page.

PHICUR - A FORTRAN variable, whose value is the displacement into the vector PAGE where the current page begins.

PREF - A FORTRAN vector, used with NDBKF by the DBCS to determine which data base page should be rewritten into the DBF.

PRH - The Physical Record Header; a data base control block which preceeds every logical record and every data base hole.

RDB - The Record Description Block; an object schema control block, which describes the logical structrue of a record.

Record (or record type) - A named collection of items. There is an arbitrary number of occurrences of logical records for each record type.

RECTAB - A FORTRAN vector, used to store the RDB and IDB.

Schema - See object schema

SDB - The Set Description Block; an object schema control block which describes the logical structure of a set.

Set (or set type) - A named collection of records which specifies an ordering or relation among the records.

SETTAB - A FORTRAN vector, used to store the SDB, ODB, and MDB.

APPENDIX A

Routine Names and Numbers for DBERR

Routine	Routine Number	Routine	Routine Number
AMS	1	SFM	34
CLOS	2	SFO	35
CMT	3	SFR	36
COT	4	SM	37
CR	5	SMK	38
CRS	6	SMM	39
DR	7	SMO	40
DRK	8	SO	41
DRM	9	SOK	42
DRO	10	SOM	43
FFM	11	SOO	44
FLM	12	SRK	45
FMSK	13	SRM	46
FNM	14	SRO	47
FPM	15	USE	48
GETK	16	CARD	49
GETM	17	CMK	50
GETO	18	CMM	51
GETR	19	CMO	52
GFK	20	CMR	53
GFM	21	COK	54
GFO	22	COM	55
GFR	23	COO	56
GKM	24	COR	57
GKO	25	FNSK	58
GKR	26	DBST	59
GTK	27	SKFM	60
GTM	28	SMFM	61
GTO	29	SOFM	62
OPEN	30	SRFM	63
RM	31	AMSK	64
RS	32	AMSM	65
SFK	33	AMSO	66

Routine	Routine Number
DELS	67
SETK	68
SETM	69
SETO	70
SETR	71
ICFK	72
ICFM	73
ICFO	74
ICFR	75