IMPLEMENTATION OF A PORTABLE DATABASE MANAGEMENT SYSTEM

Richard Omer

and

Michael J. Smith

Department of Computer and Information Sciences

University of Florida

Gainesville, Florida 32611

ABSTRACT

A solution to the problem of database software portability and interchange has been achieved by the implementation of the Automated Information Management (AIM) system. AIM, a CODASYL-type database management system, is written entirely in a high-level industry-wide standard language. The technique of implementation is carefully designed to eliminate any machine dependencies such as word size or internal data representation. Thus, a program written using AIM will run on any machine supporting a commonly available subset of ANSI 1974 COBOL.

The specifications of the Data Base Task Group, published in 1969 and 1971, and the later enhancements by the Data Definition Language Committee, published in 1973, have gained increasing acceptance by the data processing community. Although various computer manufacturers have introduced database management systems based on these proposals, each of these systems represent mutually non-compatible versions of the standard. The approach taken in the development of AIM has overcome this problem and made CODASYL database facilities available to a broader class of machines than ever before.

INTRODUCTION

Within the past decade, there has been a growing recognition of the complexities surrounding the maintenance of large masses of data. This has lead to the introduction of many software systems dedicated to the specific problems of centralized control and access to computer information files. Unfortunately, these systems are typically highly bound to the family of machines for which they were written. Because of the perceived need for a machine-independent database management system (DBMS), work on the AIM project was begun in the summer of 1974. The result has been the successful implementation of a portable DBMS based on the 1969 report of the Data Base Task Group (CODASYL committee).

This paper outlines the functional capabilities of AIM, as well as the techniques employed in its design. First, criteria are set down which lead to machine-independence; next, the operational capabilities of AIM are briefly delineated; a short discussion of data attributes and storage structures follows; concluding remarks are presented in a summary at the end. The appendix contains a sample AIM program.

IMPLEMENTATION CRITERIA

In order to meet the stated design goal of machine-independence, several decisions were made concerning the means by which the AIM system would be implemented. Primarily, an industry standard language, COBOL, was to be used exclusively in its realization. Though this choice may have an impact on the flexibility of the system, we feel that any sacrifice in processing efficiency will be minimal indeed. COBOL tends to excel in I/O intensive applications, an area which most database problems fall into.

COBOL itself is not a single language; rather, each compiler has its own unique set of enhancements, deficiencies, and eccentricities. Thus a subset of COBOL had to be selected that would be broad enough to be useful yet narrow enough to be common to most presently available compilers. We chose as our subset that portion of the language common to both the IBM OS/360 Version 4 compiler and the ANSI 1974 standard. For ease of implementation, we chose to use the Table Handling feature (SEARCH verb), the String Manipulation feature (STRING and UNSTRING verbs) and the Subroutine Linkage feature (CALL verb and LINKAGE SECTION). However, other less useful or less widely available features were avoided (such as the Sort feature and Report Writer).

The COPY statement proved to be an invaluable tool in eliminating certain areas of inherent incompatibility between different compilers. For example, the IBM compiler anticipates the key of a random access file to be named in a "NOMINAL KEY" clause, while the DECSYSTEM-10 compiler calls for the use of "ACTUAL KEY". This difficulty is overcome by means of COBOL's Source Library facility. Text can be inserted into a program at various points from a user created library of source code. Thus, in those areas where residual conflicts exist, only the COPY library needs to be changed, not the program itself. This also leads to the ability to perform global reconfiguration of AIM, such as incorporating additional system buffers or changing record sizes.

Specific attention was paid to the use of questionable data types such as COMPUTATIONAL-3 and the use of the REDEFINES clause. ANSI COBOL language specifications are inclined to leave internal representations of data to the discretion of the individual compiler writer. This does provide a wide range for variations. In

particular, an obvious pitfall would be to assume that a COMPUTATIONAL item with a given PICTURE (say S99) would be represented as either a sixteen bit half-word (as with IBM/370) or as a thirty-six bit full-word (as with DECSYSTEM-10). Furthermore, it would be false to assume that all manufacturers represent a character internally in either eight bits (such as IBM/370) or in six bits (such as DECSYSTEM-10).

OPERATIONAL CHARACTERISTICS

Use of the AIM DBMS is a three stage process. First, a description of the data items to be stored in the database and the relationships between them is prepared. This description is coded in terms of the AIM Data Definition Language (DDL). The DDL source deck is input to the File Definition Processor program (FDP) which in turn produces two output files. The schema file contains the internal table to be used by AIM during the next two phases. The copy file is a valid COBOL Data Division data structure that will be used by the application programmer as a communications and parameter passing area with AIM.

The second stage involves formatting of direct access storage space for the AIM database. This is accomplished by means of the Database Initialization program (DBINIT).

The third stage entails the use of the AIM run-time subroutines to perform the various database access functions. Subroutines are provided to handle the following actions:

* Initiate database processing (OPENDB)
* Cease database processing (CLOSEDB)
* Store a new record (STORE)
* Physically remove a record (REMOVE)
* Logically though not physically remove a record (DELETE)
* Modify the contents of an item within a record (MODIFY)
* Move a record into working storage (GET)
* Locate a record by record address (FINDD)
* Locate a record by means of a randomizing key (FINDG)
* Locate the next record within a chain of records (FINDN)
* Locate the preceding record within a chain of records (FINDP)
* Locate the unique master record within a chain of records (FINDM)
* Locate the current record within a chain of records (FINDC)
* Cause system buffers to be written back to the database (FLUSH)
* Print the contents of the communications table (SNAP)

DATA ATTRIBUTES

The schema describes the database in terms of the names and characteristics of the items, groups, and chains included in the database. An item is a data element that is not structurally subdivided and that is associated with occurrences of values. This term is similar to "elementary item" in COBOL, or "field" in a punched card. An item always has the attributes of type and length. AIM permits an item to be defined as either numeric or alpha-numeric. The length of a numeric item must be less than eighteen digits, while the length of an alpha-numeric item may be anything up to the maximum number of characters in a page.

A group is a collection of items identifiable as a unit. This term is similar to "record". Groups always have the attributes of reference-code

(logical address), synonym, and location mode. There are four possible location modes that a group might have. PRIMARY groups are located by their unique AIM assigned reference-codes (ref-code). INDIRECT groups are located through their position on a predefined chain and in relation to the unique master group in that chain. RANDOM and RANDOMX groups are identified through the value contained in a named randomization key defined within the group. The distinction between the two is that the former does not allow groups with duplicate key values to exist in the database, while the latter does permit such duplicates. The synonym of a group is merely a single character assigned by the user to act as an "abbreviation" for the group's name. Its primary use is to distinguish master from detail during a find-next operation.

Chains are ordered, logical collections of associated groups. Each chain must be named and have one master group type and one or more detail group types declared for it is the DDL. It is the function of the AIM system to maintain chains in the order specified for them. There are five possible chain orderings. SORTED order implies that detail groups are placed within the chain based upon the value in a given sort key field within that group. An order of FIRST means that each new detail group is placed as the first group relative to the master group within the chain. An order of LAST causes AIM to store a new group as the last group within the chain. Insertion orders of NEXT and PRIOR are used to specify that the new group is to be placed in a position either immediately following or else immediately preceding the group last accessed on that chain.

DATA STORAGE CHARACTERISTICS

The data file may be divided into one or more regions, representing a contiguous allocation of direct access storage space. Each region is composed of one or more pages, representing the units of physical database access. Pages are all of a single fixed length. Each page may contain zero to ninety-nine lines, each representing a particular occurrence of some group. As a group is stored in the database, it is assigned a unique seven character identification called its ref-code. This code is composed of three parts: an alphabetic region code, a four digit page number, and a two digit line number. Thus an AIM database may comprise up to 260,000 pages.

Each AIM page contains a fixed overhead of eighteen characters plus three characters for each group stored within it. The first sixteen characters form the page header and serve to identify the page number, the amount of free space available, and a pointer to the first random group stored on that page. Groups (if any) stored on the page follow the page header information. After all the groups, a checksum is appended in order to insure the integrity of the data stored there. The three character overhead associated with a group is made up of the two digit line number of the group and the one character synonym of the group. Additional overhead is consumed within a group for each chain passing through it. Since every chain always has pointers in the forward direction, at least seven characters will be set aside. If the chain also has the optional prior or head pointers declared for it in the DDL, the overhead per chain may go as high as fourteen or twenty-one characters.

As mentioned earlier, the portability criterion caused us to pay special attention to the use of data types, especially in those cases where either an explicit or implicit redefinition of one type over another might occur. This consideration is most apparent in out choice of data structures for external storage. Within a page of data as retrieved from disk, it must be possible to extract any group stored within it. If a mixture of binary and alpha-numeric

information were allowed to coexist within that group, this would raise the problem of machine dependent word size. For some machines, a binary integer may correspond to four character positions; in other machines, the correspondence is six characters. Furthermore, various computers will expect differing data alignment conditions to be met.

Two solutions to this problem are: (1) extensive parameterization of the program, to the point that these dependencies can be altered from one implementation to the next; (2) adapt the convention that all data stored on a page must be in character format. We took the second alternative as a simplier, more direct avenue for a practical implementation. This decision is reflected in the PICTURE of a ref-code (PIC X9(6)). Though the desired range of page and line numbers could have been represented more compactly as a binary integer, we would have been forced to adapt the former approach since ref-codes are used as pointers within groups and this would cause us to mix binary and alpha-numeric data within a page.

CONCLUDING OBSERVATIONS

Three factors have played an important role aiding in the development effort behind AIM: modularity of design, structured programming techniques, and the liberal use of comments and internal error messages. Needless to say, any large software system (AIM consists of over nine thousand lines), requires certain disciplines to be followed. The sectioning of the problem into manageable, well-defined modules, typically less than eight hundred lines, was quite helpful during all phases of the project. Internal error messages have also proven to be a wise practice, as they simplify the debugging phase and help insure program correctness. Last, but not least, a "structured" programming technique was used that made coding easier to follow and which encouraged each module to be broken down into further sub-modules. Though COBOL does not support the kind of language constructs that make entirely "GO TO" free programming possible, the convention of using GO TO statements in only one restricted context (GO TO the EXIT of a PERFORMed paragraph) was judged to be a very effective structuring technique.

AIM has now been operational for three months and in that time has been used as the basis of several on-line retrieval systems. During that time considerable interest in the project has been generated both within the University of Florida and elsewhere. Among some of the intriguing applications proposed has been its use in distributed database architectures spanning computers of differing manufacture. AIM is now serving as the prototype for future database development projects here at the University of Florida.

Appendix: Sample DDL and COBOL program

```
PAGE 0001            FILE DEFINITION PROCESSOR              DATE   03/11/76

 0001  DDL    LIST    NOSCHEMA
 0002  FILE   NAME=DDLEXPL   PAGES=400   CHARS=1500
 0003         PASSWORD=NOJOKE   INV=85
 0004     GROUP   NAME=FIELD   SYN=X   TYPE=PRIMARY
 0005           CHAIN   NAME=WRITERS,MASTER,PRIOR   ORDER=SORTED
 0006           CHAIN   NAME=SUBJECTS,MASTER        ORDER=NEXT
 0007     GROUP   NAME=WRITER  SYN=Y   TYPE=RANDOMX,AUTHNAME
 0008        ITEM   NAME=AUTHNAME     TYPE=A,20
 0009        ITEM   NAME=ADDRSS       TYPE=A,30
 0010        ITEM   NAME=AGE          TYPE=N,2
 0011           CHAIN   NAME=WRITERS,DETAIL          SKEY=AUTHNAME,A
 0012                   DUP=LAST
 0013           CHAIN   NAME=PAPERS,MASTER           ORDER=LAST
 0014     GROUP   NAME=SUBJECT SYN=Z   TYPE=SANDOM,SUBJAREA
 0015        ITEM   NAME=NUMPAPR      TYPE=N,4
 0016        ITEM   NAME=SUBJAREA     TYPE=A,25
 0017           CHAIN   NAME=SUBJECTS,DETAIL
 0018           CHAIN   NAME=LINKCHN,MASTER          ORDER=NEXT
 0019     GROUP   NAME=PAPER   SYN=W   TYPE=INDIRECT,PAPERS
 0020        ITEM   NAME=TITLE        TYPE=A,40
 0021        ITEM   NAME=JOURNAL      TYPE=A,40
 0022        ITEM   NAME=PUB-DATE     TYPE=N,6
 0023           CHAIN   NAME=PAPERS,DETAIL
 0024           CHAIN   NAME=LINKCHN,DETAIL,HEAD
 0025  END
TEXT PROCESSING COMPLETE  --  000  ERRORS DETECTED  --

PAGE 0002            FILE DEFINITION PROCESSOR              DATE   03/11/76

FILE NAME: DDLEXPL   SIZE IN PAGES:    400  CHARACTERS/PAGE: 1500
INVENTORY LEVEL: 85  PASSWORD: NOJOKE
GROUP TABLE:

-------------------------------------------------------------
I   NAME   I SYN I TYPE I SIZE I  LOC  I   KEY    I  CHAINS  I
-------------------------------------------------------------
I          I     I      I      I       I          I          I
I FIELD    I  X  I  PR  I  31  I   70  I          I WRITERS  I
I          I     I      I      I       I          I SUBJECTS I
I          I     I      I      I       I          I          I
I WRITER   I  Y  I  RX  I  90  I  147  I AUTHNAME  I WRITERS  I
I          I     I      I      I       I          I PAPERS   I
I          I     I      I      I       I          I          I
I SUBJECT  I  Z  I  RN  I  53  I  241  I SUBJAREA  I SUBJECTS I
I          I     I      I      I       I          I LINKCHN  I
I          I     I      I      I       I          I          I
I PAPER    I  W  I  IN  I 117  I  312  I PAPERS    I PAPERS   I
I          I     I      I      I       I          I LINKCHN  I
I          I     I      I      I       I          I          I
-------------------------------------------------------------

PAGE 0003            FILE DEFINITION PROCESSOR              DATE   03/11/76

CHAIN TABLE:

-----------------------------------------------------------------------
I   NAME   I ORDER  I PRIOR I HEAD I  MASTER  I  DETAIL  I  LOC  I   KEY    I
-----------------------------------------------------------------------
I          I        I       I      I          I          I       I          I
I WRITERS  I SORT-2 I  YES  I NO   I FIELD    I WRITER   I   77  I AUTHNAME I
I          I        I       I      I          I          I       I          I
I SUBJECTS I NEXT   I  NO   I NO   I FIELD    I SUBJECT  I  112  I          I
I          I        I       I      I          I          I       I          I
I PAPERS   I LAST   I  NO   I NO   I WRITER   I PAPER    I  206  I          I
I          I        I       I      I          I          I       I          I
I LINKCHN  I NEXT   I  NO   I YES  I SUBJECT  I PAPER    I  277  I          I
I          I        I       I      I          I          I       I          I
-----------------------------------------------------------------------

PAGE 0004            FILE DEFINITION PROCESSOR              DATE   03/11/76

ITEM TABLE:

-----------------------------------
I   NAME   I  LOC  I TYPE I LENGTH I
-----------------------------------
I          I       I      I        I
I AUTHNAME I  154  I  AN  I   20   I
I          I       I      I        I
I ADDRSS   I  174  I  AN  I   30   I
I          I       I      I        I
I AGE      I  204  I  NU  I    2   I
I          I       I      I        I
I NUMPAPR  I  248  I  NU  I    4   I
I          I       I      I        I
I SUBJAREA I  252  I  AN  I   25   I
I          I       I      I        I
I TITLE    I  319  I  AN  I   40   I
I          I       I      I        I
I JOURNAL  I  359  I  AN  I   40   I
I          I       I      I        I
I PUB-DATE I  399  I  NU  I    6   I
I          I       I      I        I
-----------------------------------
```

```
PP 5734-CB2 V4 RELEASE 1.2 31OCT73       IBM OS AMERICAN NATIONAL STANDARD COBOL

  1
00001          IDENTIFICATION DIVISION.
00002          PROGRAM-ID.  EXPLPGM.
00003          ENVIRONMENT DIVISION.
00004          DATA DIVISION.
00005          WORKING-STORAGE SECTION.
00006          77  GROUP-NAME      PICTURE X(8).
00007          77  CHAIN-NAME      PICTURE X(8).
00008          01  COM-TAB         COPY DDLEXPL.
00009 C        01  COM-TAB.
00010 C            05  CCB.
00011 C                09  REF-CODE            PIC IS X9(6).
00012 C                09  GRP-SYN             PIC IS X.
00013 C                09  ERR-CODE            PIC IS S999.
00014 C                09  ERR-SYN             PIC IS X.
00015 C                09  ERR-REF             PIC IS X9(6).
00016 C                09  PASSWORD            PIC IS X(8).
00017 C            05  PAGEHDR.
00018 C                09  CURR-OF-PAGE        PIC IS X9(6).
00019 C                09  CALCHAIN.
00020 C                    13  CHN-MSTR        PIC IS X9(6).
00021 C                    13  CHN-PRIR        PIC IS X9(6).
00022 C                    13  CHN-CURR        PIC IS X9(6).
00023 C                    13  CHN-NEXT        PIC IS X9(6).
00024 C                    13  CHN-GRP         PIC IS X9(6).
00025 C            05  FIELD.
00026 C                09  CURR-OF-X           PIC IS X9(6).
00027 C                09  WRITERS.
00028 C                    13  CHN-MSTR        PIC IS X9(6).
00029 C                    13  CHN-PRIR        PIC IS X9(6).
00030 C                    13  CHN-CURR        PIC IS X9(6).
00031 C                    13  CHN-NEXT        PIC IS X9(6).
00032 C                    13  CHN-GRP         PIC IS X9(6).
00033 C                09  SUBJECTS.
00034 C                    13  CHN-MSTR        PIC IS X9(6).
00035 C                    13  CHN-PRIR        PIC IS X9(6).
00036 C                    13  CHN-CURR        PIC IS X9(6).
00037 C                    13  CHN-NEXT        PIC IS X9(6).
00038 C                    13  CHN-GRP         PIC IS X9(6).
00039 C            05  WRITER.
00040 C                09  CURR-OF-Y           PIC IS X9(6).
00041 C                09  AUTHNAME            PIC IS X(20).
00042 C                09  ADDRSS              PIC IS X(30).
00043 C                09  AGE                 PIC IS S9(2).
00044 C                09  PAPERS.
00045 C                    13  CHN-MSTR        PIC IS X9(6).
00046 C                    13  CHN-PRIR        PIC IS X9(6).
00047 C                    13  CHN-CURR        PIC IS X9(6).
00048 C                    13  CHN-NEXT        PIC IS X9(6).
00049 C                    13  CHN-GRP         PIC IS X9(6).
00050 C            05  SUBJECT.
00051 C                09  CURR-OF-Z           PIC IS X9(6).
00052 C                09  NUMPAPR             PIC IS S9(4).
00053 C                09  SUBJAREA            PIC IS X(25).
00054 C                09  LINKCHN.
00055 C                    13  CHN-MSTR        PIC IS X9(6).
00056 C                    13  CHN-PRIR        PIC IS X9(6).
00057 C                    13  CHN-CURR        PIC IS X9(6).
00058 C                    13  CHN-NEXT        PIC IS X9(6).
00059 C                    13  CHN-GRP         PIC IS X9(6).
00060 C            05  PAPER.
00061 C                09  CURR-OF-W           PIC IS X9(6).
00062 C                09  TITLE               PIC IS X(40).
00063 C                09  JOURNAL             PIC IS X(40).
00064 C                09  PUB-DATE            PIC IS S9(6).
00065          PROCEDURE DIVISION.
00066              MOVE "NOJOKE" TO PASSWORD.
00067              CALL "OPENDB" USING COM-TAB.
00068              ACCEPT AUTHNAME.
00069          * LOCATE THE RECORD DESCRIBING AUTHOR AND HIS PAPERS
00070              MOVE "WRITER" TO GROUP-NAME.
00071              CALL "FINDG" USING COM-TAB GROUP-NAME.
00072          * NOW RETRIEVE ALL PAPERS WRITTEN BY GIVEN AUTHOR.
00073              MOVE SPACE TO GRP-SYN.
00074              PERFORM NEXT-PAPER THRU NEXT-PAPER-X
00075                UNTIL GRP-SYN = "Y".
00076              CALL "CLOSEDB" USING COM-TAB.
00077              STOP RUN.
00078          NEXT-PAPER.
00079          * DO A FIND-NEXT TO LOCATE NEXT PAPER ON PAPERS CHAIN
00080              MOVE "PAPERS" TO CHAIN-NAME.
00081              CALL "FINDN" USING COM-TAB CHAIN-NAME.
00082          * CHECK GRP-SYN. "Y" MEANS NO MORE PAPERS REMAIN
00083              IF GRP-SYN = "Y"
00084                GO TO NEXT-PAPER-X.
00085          * MOVE THE RECORD INTO WORKING-STORAGE.
00086              MOVE "PAPER" TO GROUP-NAME.
00087              CALL "GET" USING COM-TAB GROUP-NAME.
00088          * ALSO GET THE SUBJECT GROUP ASSOCIATED WITH THIS PAPER
00089              MOVE "LINKCHN" TO CHAIN-NAME.
00090              CALL "FINDM" USING COM-TAB CHAIN-NAME.
00091              MOVE "SUBJECT" TO GROUP-NAME.
00092              CALL "GET" USING COM-TAB GROUP-NAME.
00093          * FINALLY, DISPLAY THE RESULTS FOR CURRENT PAPER.
00094              DISPLAY TITLE SUBJAREA JOURNAL PUB-DATE.
00095          NEXT-PAPER-X.
00096              EXIT.
```