QUAESTIONES INFORMATICAE

Vol. 1 No. 2

September, 1979

Quaestiones Informaticae

An official publication of the Computer Society of South Africa
'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Afrika

Editors: Dr. D. S. Henderson,
Vice Chancellor, Rhodes University, Grahamstown, 6140, South Africa.
Prof. M. H. Williams,
Department of Computer Science and Applied Maths,
Rhodes University, Grahamstown, 6140, South Africa.

Editorial Advisory Board

Subscriptions

Annual subscriptions are as follows:

	SA	US	UK
Individuals	R2	$3	£1.50
Institutions	R4	$6	£3.00

Quaestiones Informaticae is prepared for publication by SYSTEMS PUBLISHERS (PTY) LTD

for the Computer Society of South Africa.

Direct Fortran/IDMS Interface (without the use of a DML) on the ICL 1904/2970 Computers, using a geological data base as example.

Barbara Day
Gold Fields of South Africa Limited

Abstract

The Integrated Data Base Management System (IDMS) originally had only a COBOL data manipulation language (DML). Recently a DML for FORTRAN on the 2900 series of ICL computers has been produced. This is a piece of software that has to be purchased as an "optional extra" once the IDMS package has been obtained; this may dissuade potential FORTRAN users, particularly if FORTRAN is only to be used infrequently.
This paper describes a simple method of using IDMS directly from a FORTRAN program, without the use of any DML. The method is described for both the ICL 1904 and 2970 machines. In the description reference is made to the application for which the method was first developed, viz. a geological data base for Gold Fields of South Africa Limited.

1. Background

Integrated Data Base Management System (IDMS) is a software package which handles the establishment and processing of a data base [1]. It was originally written by the B.F. Goodrich Company of Akron, Ohio in 1971. In 1973 the Cullinane Corporation of Boston, Massachusetts acquired marketing and development rights; these rights were later obtained for their own equipment by ICL.
Using IDMS, data can be stored in a network structure; to actually store and access this data application programs have to be written. Those in COBOL have the usual COBOL language supplemented by several data manipulation verbs which deal with storage/access of data on the data base. This enhanced COBOL is referred to as the COBOL data manipulation language (COBOL DML).
The FORTRAN DML is purchased or hired separately from the usual IDMS package.
At Gold Fields of South Africa Limited (GFSA), it was decided early in 1978 to develop a geological data base system to deal with ore reserve calculations, mine planning and to assist with many geological/metallurgical problems for a new base metal mining project being developed in the north western Cape. Generally computer systems at GFSA are developed in COBOL. With this project, however, it was felt that since certain of the programs involved fairly complex mathematics, they could be more efficiently written in FORTRAN. Early in 1978 a pilot project for the data base was developed on the ICL 1904S. Included in this project was a FORTRAN program. With the aid of ICL's Software Products Department, headed by Mr. Michael Wilson and documentation from the Water Data Unit in Reading, United Kingdom, [2] a direct FORTRAN/IDMS interface was developed. Later in 1978 GFSA acquired an ICL 2970 computer and work commenced on the "live" geological data base. Most of the initial work dealt with data vetting procedures, load programs, prints and data base checking routines, all written in COBOL. Later FORTRAN programs were developed and the 1900 FORTRAN/IDMS interface was modified to suit the 2970 machine.

2. IDMS/FORTRAN Interface on the 1904S computer

2.1. Restrictions

With the ICL 1900 computers all FORTRAN variables are aligned on word boundaries. This must be considered when attempting to communicate with an IDMS data base in a FORTRAN program.
In the GFSA application all numeric variables were treated as integers. It was found that the method of indicating a decimal point position with a "V" in the schema was not picked up by the FORTRAN program (e.g. PICTURE 9V999 would be picked up as a 9999 integer). All numeric fields were stored and read from the data base as whole numbers; suitable division was done to yield the required number of decimal digits.

2.2 Basis of Method

The data base is accessed by means of CALL statements in the FORTRAN programs. These take the form:

CALL IDMS (parameter,parameter – – – –)

The parameters will be discussed below in detail. The first parameter is always a code indicating what is to be done to the data base. This code is always a 4-digit integer and the parameter must always be the integer name not the constant itself or even an array element. The Water Data Unit (Reading) discovered that array elements could not be passed as parameters and this was acknowledged as a bug by the ICL IDMS implementation team. Array elements had to be equivalenced to single variables before being used as parameters. The first digit of this first parameter always indicates the total number of parameters to be used in the call e.g. a code of 2041 indicates that there should be a total of 2 parameters in the CALL statement. Other parameters depend on the action required and include record and set names; details are found in the following sections.

In order to use the data base it is necessary to set up storage areas for use by IDMS and to initialize certain fields including the first parameter described above; furthermore the subschema must be explicitly described. All this is done in a block data segment.

2.3 Detailed Description of the FORTRAN/IDMS Interface

2.3.1 The Block Data Segment

A block data segment must be written using the items shown in Table 1. This segment defines the subschema to be used in the FORTRAN program. It also defines IDMS control areas which can be accessed by statistical procedures to indicate the progress of a program.

The coded block data segment consists of COMMON statements dealing with each item defined in Table 1. DATA statements deal with the initialization of those items that must have certain values assigned to them.

2.3.2 Making the Subschema Available to the Master Segment

Before accessing the data base it is necessary to "BIND" in the subschema e.g.:-

CALL IDMS (IBNDRU,PNAME,SSN)

Where IBNDRU has been initialized to 3059 in the block data subprogram,

PNAME has been initialized to its program name,

SSN is the variable containing the subschema name.

Further it is necessary to "BIND" in each record type:

CALL IDMS (IBNDRC,S1,I1)

where IBNDRC has been initialized to 3048 in the block data segment. S1 is the variable name of the record found in the subschema record names area. I1 is the name given to the first word in the record. This field should be equivalenced to a single variable if previously defined as an array element.
As a result, the records of the subschema and their contents become available to the program.

2.3.3 Obtaining Access to the Data Base

Even after the BINDS in 2.3.2 records cannot be accessed until the data base has been "READIED".
Two examples are given here:-

CALL IDMS (ICODE) ICODE = 1037

which readies the data base with retrieval usage and

CALL IDMS (ICODE, AREA1) ICODE = 2041

AREA1 contains an area name which readies AREA1 with exclusive update.

2.3.4 Using the Data Base

The above examples explain how direct calls are made to the data base software. All the usual functions are possible (e.g. retrieval, modification, storage, erasing, etc. of records). In each case it is necessary to know the required code and other call parameters. Table 2 gives a subset of the possible ways of accessing the data base.

TABLE 1

BLOCK DATA SEGMENT

Subschema Control Area

Field	Size	Initialization
Program name	8 characters	Program name
Error Status	4 characters	"1400"
Data base key	1 word	
Record and realm names	16 characters	Spaces
Direct data base key	1 word	
Record occurrence	1 word	
DML sequence	1 word	

Statistical Work Area

IDMS date	8 characters
IDMS time	8 characters
Pages read ex-data base	1 word
Pages written to data base	1 word
Pages reqested by DBMS	1 word
Calc. records that hit in page	1 word
Calc. records that overflow	1 word
Via records in owner page	1 word
Via records in overflow	1 word
Lines requested by DBMS	1 word
Records became current of run unit	1 word
Call to DBMS	1 word

Name	Size	Initialization
Subschema name	8 characters	Actual subschema name
Record, set and realm names	16 characters	Use actual names in subschema

N.B. The first record name must be "SR1▽▽▽▽▽▽▽▽▽▽▽▽▽"
The last set name must be "CALC▽▽▽▽▽▽▽▽▽▽▽▽▽"

Record Buffer Areas and IDMS Codes

Fields to hold the contents of each record in the subschema must be set up without initialization.

All IDMS codes to be used must be set up and initialized.

An OBTAIN statement can be used with any of the above expressions below by adding 1 to the *first digit* of parameter 1, and using an additional *last* parameter of 0043 e.g. OBTAIN CALC record-name becomes

OBTAIN CALC record-name	3032	RN	0043

To GET a record:-

GET	1043	
GET record-name	2034	RN

To actually *read* a record either an OBTAIN must be used or a FIND followed by a GET. The data is then to be found in the appropriate fields of the record buffer area set up in the block data subprogram.

TABLE 2

RN = record name as set up in block data segment
SN = set name
AN = area name

Statement	Parameter 1	Parameter 2	Parameter 3 & 4
FIND DB-KEY is identifier	2005	identifier	
FIND record-name DB-KEY is identifier	3006	RN	identifier
FIND CURRENT	1030		
FIND CURRENT record-name	2007	RN	
FIND CURRENT WITHIN set-name	2008	SN	
FIND CURRENT WITHIN area-name	2009	AN	
FIND NEXT WITHIN set-name	2014	SN	
FIND NEXT record-name WITHIN set-name	3010	RN	SN
FIND PRIOR WITHIN set-name	2016	SN	
FIND PRIOR record-name WITHIN set-name	3012	RN	SN
FIND FIRST WITHIN set-name	2020	SN	
FIND OWNER WITHIN set-name	2031	SN	
FIND CALC. record-name	2032	RN	
FIND DUPLICATE record-name	2050	RN	
STORE	2042	RN	
CONNECT	3044	RN	SN
MODIFY	2035	RN	
DISCONNECT	3046	RN	SN
ERASE	2052	RN	
ERASE PERMANENT	2003	RN	
ERASE SELECTIVE	2053	RN	
ERASE ALL	2004	RN	
ACCEPT identifier FROM set-name OWNER CURRENCY	3067	SN	identifier
ACCEPT identifier FROM IDMS-STATISTICS	2066	identifier	
IF set-name is EMPTY . . .	2064	SN	
IF set-name IS NOT EMPTY . . .	2065	SN	
IF set-name MEMBER . . .	2060	SN	
IF NOT set-name MEMBER . . .	2062	SN	

After the last four calls it is necessary to use a normal FORTRAN IF statement to test the contents of error status (in the subschema control area of the block data subprogram). If the set is empty then error status equals '0000'; if not empty status contains '1601'. For the last two statements error status contains '0000' if the record (current of run-unit) is linked into the specified set; '1601' if it is NOT a member.

2.3.5 IDMS-Routines

In a normal IDMS application program the following standard routines are used:-

IDMS-STATISTICS which gives a statistical analysis of the accesses to the data base, printing out the items listed in the statistical work area of the block data segment (see Table 1).

IDMS-STATUS checks the error-status (see subschema control area in Table 1) and if this is non-zero this indicates an error condition and a call is made to IDMS-ABORT, where the program is aborted after a FINISH statement has been executed.

These routines can readily be coded in FORTRAN and additional error detection statements can be incorporated.

3. FORTRAN/IDMS Interface on the 2970

3.1 Comparison with the method on the 1904

Since the 2970 is a byte machine rather than a word machine the condition of all variables commencing on a word boundary was not as critical. However to avoid any possible problems the schema was defined such that all integer variables did start on a word boundary. However the ability to define the length of character variables in 2970 FORTRAN made the use of the method simpler. The method was basically the same in that CALL's were made to IDMS routines. The format of the call statement differed viz:-

CALL ICL9IDMS (parameter,- - - - - -)

Again the first parameter is a code indicating what is to be done to the data base. However in this case this parameter is always an array element and it is the address of this element that indicates what is required. Consequently these array elements do not have to be initialized. However this does mean that extra care has to be taken in setting up the block data segment so that the correct address can be assigned to this first parameter array.

3.2 Block Data Segment

Table 3 explains the contents of the block data segment for the 2970. In 2970 FORTRAN integer variables can be specifically declared to be one or two words in length.

In the block data segment all IDMS control areas, data areas, records and sets are defined in COMMON statements. The length of each item is then carefully defined in CHARACTER and INTEGER statements. Initial values are given in DATA statements. This block data segment can be compared with the working storage section of an expanded COBOL/IDMS program. (The expanded listing is produced after the user written program goes through the IDMS pre-processor). With a FORTRAN program the user has to specifically write the segment if this method of no DML is adopted while with COBOL it is automatically handled by the software. In actual use at GFSA one block data segment was written comprising a subschema which covered the total schema; this was then accessed by all the FORTRAN programs.

3.3 Codes

The first parameter in the CALL statements is always an element from the IDMS — communications array; the address of the element indicates the action required.

TABLE 3

Block Data Segment
In this subprogram areas to be used by IDMS are allocated and initialized.

Subschema control area — which contains:-

Field	Size	Initialization
Program name	8 characters	Program name
Error status	4 characters	"1400"
Data base key	4 byte integer	
Record name	16 characters	Spaces
Realm name	16 characters	Spaces
Error set	16 characters	Spaces
Error record	16 characters	Spaces
Error realm	16 characters	Spaces
IDMS communication area	Array of size 100; each element being a character of length length 1 byte	
Direct data base key	Integer of 4 bytes	
TP phase ID	Integer of 8 bytes	
IDMS target area	16 characters	
A filler	24 characters	
DB statement code	2 characters	
DB status code	5 characters	
A filler	1 character	
Record occurences	Integer of 4 bytes	
DML sequence	Integer of 4 bytes	

Statistics work area — which contains:-

IDMS date	8 characters
IDMS time	8 characters
Pages read from data base	4 byte integer
Pages written to date base	4 byte integer
Pages requested by DBMS	4 byte integer
Calc. records that hit in page	4 byte integer
Calc. records that overflowed	4 byte integer
Via records same page as owner	4 byte integer
Lines requested by DBMS	4 byte integer
Records became current of run unit	4 byte integer
Calls to DBMS	4 byte integer
Fragments stored	4 byte integer
Records relocated	4 byte integer
"PROVERFLOW"	4 byte integer
"PDSAME"	4 byte integer
"PDOVER"	4 byte integer
Filler	36 characters

None of the above fields is initialized.

Month-Date-Time Area

Month name abbreviations	36 characters
Date-format 1	9 characters
Date-format 2	8 characters
Time	11 characters

These fields could be initialized if time and dates are to be printed.

The subschema name, record, set and realm names as well as the record buffer areas are set up exactly as for the 1904.

3.4 Compilation and Collecting of Programs
Programs are compiled in the normal way as for any non-IDMS FORTRAN program. There is no need to specifically collect the IDMS routines into the object code as is required on the 1900 machines.

3.5 Obtaining the Subschema and Gaining Access to the Records
As explained for the 1904 procedure it is necessary to BIND in the subschema and each record type. At GFSA this was done by writing a subroutine containing the necessary CALL statement to IDMS. This subroutine was then called in by each FORTRAN program.
To compare the method of binding in the subschema on the 1904 and the 2970 the following IDMS calls are shown:-

1904: CALL IDMS (IBNDRU,PNAME,SSN) where IBNDRU = 3059

2970: CALL ICL9IDMS (IDMSCOM(59),PNAME,SSN) where IDMSCOM(59) is the 59th element of the IDMS communication array set up in the Block Data Segment (Table 3).

To access the data base it must first be READIED.
e.g. CALL ICL9ICMS (IDMSCON(41),AREA1) will ready area 1 for exclusive update.

3.6 Comparison with the COBOL DML
As an example an OBTAIN statement will be examined. In COBOL the programmer will code:

```
OBTAIN NEXT SAMPLE WITHIN BH-SAMP.
```

This will read into working store the next available SAMPLE record within the set BH-SAMP. The IDMS pre-processor will transform the above statement to a comment and translate it to:

```
CALL "ICL9IDMS" USING IDBMSCOM (10)
                      SR103
                      BH-SAMP
                      IDBMSCOM (43)
```

The FORTRAN equivalent is:

```
CALL ICL9IDMS(IDMSCOM(10),SAMPLE,BHSAMP,
IDMSCOM(43))
```

Clearly there is no need to use the IDMS pre-processor prior to normal FORTRAN compilations.

4. Conclusions
There is obviously some more work involved in directly accessing a data base with FORTRAN. Only by considering the frequency of usage and the actual number of man-hours involved can judgement be made if this time should be spent rather than the financial outlay on the direct FORTRAN DML (if such is available for the machine in question). The setting of work standards can greatly reduce the amount of labour, for example, the sharing of block data segments, statistical routines and BINDS subroutines by all FORTRAN/IDMS programs. This presupposes that these programs all use the same subschema. A comment, giving the IDMS statement in COBOL terms before each call to IDMS enhances the readibility of the FORTRAN program.

Acknowledgements

My thanks go to:
The Management Services Division, GFSA, for permission for this paper to be submitted to the 1st S.A. Computer Symposium; Mr. M. Wilson and the Software Products Department of ICL, Johannesburg for assistance with developing the pilot IDMS/FORTRAN interface.

References

[1] IDMS — Technical Overview — ICL
[2] Report on the Use of IDMS from FORTRAN by Water Data Unit, Reading, U.K.

Notes for Contributors

The purpose of this Journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles, exploratory articles or articles of general interest to readers of the Journal. The preferred languages of the Journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be in double-spaced typing on one side only of A 4 paper and submitted to Dr. D. S. Henderson or Prof. M. H. Williams at

Rhodes University
Grahamstown 6140
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. The original ribbon copy of the typed manuscript should be submitted. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name, and the affiliation and address. Each paper must be accompanied by a summary of less than 200 words which will be printed immediately below the title at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Illustrations and tables should not be included in the text, although the author should indicate the desired location of each in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Illustrations should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the Author's name and figure number. Original line drawings (not photoprints) should be submitted and should include all relevant details. Drawings, etc., should be about twice the final size required and lettering must be clear and "open" and sufficiently large to permit the necessary reduction of size in block-making.

Where photographs are submitted, glossy bromide prints are required. If words or numbers are to appear on a photograph, two prints should be sent, the lettering being clearly indicated on one print only. Computer programs or output should be given on clear original printouts and preferably not on lined paper so that they can be reproduced photographically.

Figure legends should be typed on a separate sheet and placed at the end of the manuscript.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters between the letter O and zero; between the letter l, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetical order of author's name, and cited in the text by number in square brackets. Journal references should be arranged thus:

1. ASHCROFT, E. and MANNA, Z. (1972). The Translation of 'GOTO' Programs to 'WHILE' Programs, in *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255.
2. BÖHM, C. and JACOPINI, G. (1966). Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM,* **9,** 366-371.
3. GINSBURG, S. (1966). *Mathematical Theory of Context-free Languages,* McGraw Hill, New York.

Proofs and reprints

Galley proofs will be sent to the author to ensure that the papers have been correctly set up in type and not for the addition of new material or amendment of texts. Excessive alterations may have to be disallowed or the cost charged against the author. Corrected galley proofs, together with the original typescript, must be returned to the editor within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Fifty reprints of each article will be supplied free of charge. Additional copies may be purchased on a reprint order form which will accompany the proofs.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.

Hierdie notas is ook in Afrikaans verkrygbaar.

Questiones Informaticae

Partial proceedings of the first South African Computer Symposium on Research in Theory, Software, Hardware, organised by The Research Symposium Organising Committee of The Computer Society of South Africa. 4 & 5 September 1979, Pretoria.

Contents/Inhoud