Guidance System for Structuring or Restructuring of a Database in Multiple Database Management Systems

YUKUO ISOMOTO*, TAKAKO MATSUDA** and NOBUYUKI TANAKA**

In a scientific information system, scientists who are usually non-professional database administrators have been collecting and editing a number of items of scientific information in ordinary file systems or handy storage-oriented DBMSs. As masses of the information materials are stored, they organize and integrate the separated ones into another large-scale database managed by an online information retrieval-oriented DBMS. Then, the scientists often have to manipulate an unfamiliar DBMS by themselves, though they are untrained. This paper addresses an experimental attempt of a guidance system with which even a non-professional database administrator can accomplish his database structuring on a DBMS unfamiliar to him.

This system is implemented as a command system named 'DBGUIDE', which provides two functions as follows: (1) the semi-automatic database conversion from COOD (a handy storage-oriented DBMS) into INQ (a general-purpose DBMS), and (2) database structuring guidance for administrators who are unfamiliar with INQ. This guidance system not only plays an important role in the management of a complicated database system in an environment where multiple databases and multiple DBMSs exist, but also shows one of the methods to familialize untrained database administrators like scientists with a DBMS.

1. Introduction

For the organization of scientific information, scientists have been integrating their specialized information into a scientific information system. Each participant in the system performs individually his database structuring in his own field with the use of the most suitable DBMS for his objects.

At an early stage in the database management, scientists who are non-professional and untrained database administrators (DBAs) collect and store the information into their private databases[1,2,3] with the use of a personal handy manipulative DBMS (or an ordinary file system). After collecting masses of the information, it is necessary to merge their databases with a high-performance DBMS for a public on-line information service[1,4]. In this case, they have to migrate their databases from a DBMS into another one.

This paper is concerned about the guidance system for database structuring or restructuring in such a situation. From the viewpoints of human factors, the system is implemented as a command system named 'DBGUIDE' in a TSS mode. DBGUIDE supports database structuring on a DBMS 'INQ'[5] which is suitable to an information retrieval-oriented, large-scale database. When a DBA has a database on a DBMS 'COOD'[6,7], it can be semi-automatically converted from COOD to INQ.

*Osaka University Computation Center, Osaka University, Mihoga-oka, 5-1, Ibaragi, Osaka, 567, JAPAN.
**Computer Center, Tohoku University, Katahira 2-1-1, Sendai, 980, JAPAN.

Recent advancement of computer technologies enables us to discuss such heterogeneous database systems with multiple DBMSs like federated or distributed database systems[8,9,10] in association with computer networks. Owing to such complicated situations, unfortunately, untrained DBAs like scientists can not perform their database structuring without assistance, though their participation is important in the formation of a scientific information system. Therefore, the guidance system improves the situation by assisting their database structuring in a conversational mode. Since this paper concentrates its subject on the guidance for database structuring, hardware and software architecture of a DBMS is beyond the scope of this paper.

2. Command System

In order to realize intelligible man-machine interactions between a DBA and a DBMS, it is very important to improve its human factors aspect[11]. Especially for the widespread dissemination of a DBMS to nontechnically trained individuals, we concentrate our discussion on the improvement of man-machine interactions between an untrained DBA and a DBMS. In this paper, man-machine interactions are improved by standardization of the procedure of database structuring.

For the standardization of database structuring, its job is divided into five steps: (1) the definition of an internal schema, (2) the definition of a conceptual schema, (3) the definition of an external schema, (4) conversion of source data, and (5) data storage to a target database. DBGUIDE executes these job steps in this order (see Fig. 1), but rejects illegal job sequences to avoid erroneous operations. Another aim of DBGUIDE

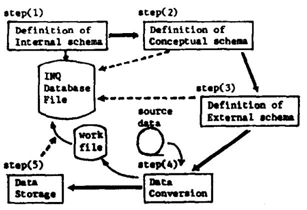


Fig. 1 Job sequence for structuring of an INQ database. DBGUIDE manages a database structuring according to this diagram which shows job sequence, data flows, and definitions. The arrows show
$\Rightarrow$; job sequence,
$\longrightarrow$; data flows,
$\dashrightarrow$; definitions of a database structure.

is an automatic database conversion from COOD to INQ. When source data are stored in a COOD database, a conceptual schema of COOD namely DDL is automatically translated into a target one of INQ namely FDL. At the same time, a data conversion program from COOD to INQ is also automatically generated in the consistent manner with the conceptual schema translation.

A DBA has access to DBGUIDE by a command 'DBGUIDE' on a TSS terminal (see Fig. 2). Following the welcome message, DBGUIDE requires a macro-name of a DBA's reserved file area, a database name, an FDL name with its sequential number, and a DBA's

```
SYSTEM ?DBGUIDE

**************************
* WELCOME TO DBS GUIDE *
**************************

UMC OF DATABASE ?
=TRACISFILE          ① A macro name of the reserved file area.

DATABASE NAME ?
=TESTDB              ② A database name.

FDL NAME AND ITS NUMBER ?
=TESTFDL 01          ③ A FDL name and its number.

USERID$PASSWORD FOR THE DATABASE ADMINISTRATOR ?
=600082040S$[redacted]  ④ An administrator's number
                                    and password.
THE DATABASE TRACISFILE/TESTDB   NEW OR OLD ?
=OLD                 ⑤ An identification of the old version
                       to be structured.
=== LIST OF THE JOB STEPS ===
  (1) INTERNAL SCHEMA           (4) DATA CONVERSION PROGRAM
  (2) CONCEPTUAL SCHEMA         (5) DATA STORAGE
  (3) EXTERNAL SCHEMA           (6) JOB HISTORY

THE NUMBER OF JOB STEP ?
=E4                  ⑥ Selection of editing mode of No. 4.

GENERATION OF DATA CONVERSION PROGRAM.

ORIGINAL DATA FILE ?
=COOD                ⑦ A name of source data.

CATA/FILE OF COOD DATABASE ?
=6000810003/EMDDL
                     ⑧ A name of a source data file.
TABLE NAME ?
=TEXTS               ⑨ A table name of a COOD database.
COOD ANALYSIS IS OK.

THE NUMBER OF RECORD SETS ?
=1000                ⑩ The maximum number of stored records.

 *** A DATA CONVERSION PROGRAM HAS BEEN GENERATED ***
```

Fig. 2 An example of man-machine interaction on DBGUIDE. User's commands are typed in along the sequence as ①, ②, · · ·, ⑩. In this figure, underlined strings are user's commands. Following the welcome message, DBGUIDE identifies a database by user's commands ①–⑤. After the identification, a user (an administrator) selects one of the six job steps. In this figure, DBGUIDE generates automatically a data conversion program by the command E4 at ⑥. After finishing the job steps, the man-machine interaction comes back to step ⑥.

number with a password in order to identify a database under structuring. Responding to the command NEW at the command input ⑤ in Fig. 2, a database is defined as new, and its relative files are prepared for the database structuring hereafter (see Table 1). On the other hand, the command OLD identifies old file which has already been defined partially. When he restarts the database structuring of old file from the other day, he has to type the command OLD at the command input ⑤.

After listing the standardized job steps, a DBA types one of them with its mode, which is an editing ('E'), run ('R'), or display mode ('D'). In this example, a DBA selects an editing mode of No. 4, namely 'DATA CONVERSION PROGRAM'. The first character 'E' specifies the editing mode. If 'R' is typed instead of 'E', the job deck will be executed on a host computer. If 'D' is typed, its corresponding job deck is displayed on a TSS terminal.

Herein, source data are assumed to be stored in a COOD database, and then a data conversion program is automatically generated on the file 'CONVPROG' (see Table 1). A COOD database is identified at the command inputs ⑦–⑩. After finishing a job step, the man-machine interactions come back to the command input ⑥. If he pushes only a carriage return key at the stage ⑥, the guidance is finished. The other job steps are also executed in a similar way.

A DBA may want to make a job schedule. DBGUIDE provides an optional function that enables a DBA to make a flexible job schedule by typing serially some of the job steps on the same line at the command input ⑥. On the one hand, if he wants to execute only one job step at a time, he only has to type the one.

3. Modules for Database Structuring Guidance

In order to support even conversational database access, it is necessary to separate the database manipulation from the other aspects of the language processor[12]. In a similar way, we separate the database structuring from the other aspects of the job execution on a host computer.

Table 1 Contents of files for DBGUIDE. The files in this table are created and initialized in the beginning of a database structuring. The contents are edited or generated by the modules of DBGUIDE.

Files	Contents
JOBHISTORY	Names for database identification. History of job execution.
INTERNAL	A job deck for the definition of an internal schema.
CONCEPTUAL	A job deck for the definition of a conceptual schema.
EXTERNAL	A job deck for the definition of an external schema.
CONVPROG	A job deck for data conversion.
STORAGE	A job deck for data loading of mass data.

Table 2 Modules of the guidance system DBGUIDE.

Modules	Functions
Supervisory module	Guides and controls the database structuring
INITIAL	Creates and initializes the files for database structuring. JCLs are created by this module.
INTERNAL	Defines an internal schema.
CONCEPTUAL	Defines a conceptual schema. For COOD, there is an option to translate automatically a DDL of COOD into an FDL of INQ.
EXTERNAL	Defines an external schema.
CONVERSN	Edit a data conversion program. For COOD, there is an option to automatically generate a data conversion FORTRAN program from COOD to INQ.
STORAGE	Automatically generates a job deck for the utility of data loading of mass data.

Taking account of the job sequence in Fig. 1, DBGUIDE consists of seven modules (see Table 2), each of which assists a DBA to edit and execute semi-automatically the job decks of his database structuring. The file JOBHISTORY stores their common information; a database name, an FDL name, a DBA's number with his password, and a job history. Referring to JOBHISTORY, each specialized module performs its work consistently with the others. In this section, let us view the individual modules of DBGUIDE.

3.1 Supervisory Module

At the beginning of the database structuring, the supervisory module identifies a target database through the command inputs ①–⑤ in Fig. 2. Following the identification, the module controls a job sequence by referring to JOBHISTORY. For an editing mode, the module calls a corresponding specialized module (see Table 2), but an illegal command or an incorrect job sequence is rejected to avoid erroneous operations.

3.2 Initialization: INITIAL

Responding just to the command NEW in the beginning, the module INITIAL creates automatically the six files listed in Table 1 in preparation for the structuring of a new database. Following the file creation, the module generates JCL sets of individual job decks on the corresponding files.

3.3 Description of Schema: INTERNAL, CONCEPTUAL, and EXTERNAL

The structure of a database is generally described in three phases: an internal, a conceptual, and an external schema. A conceptual schema of INQ consists of independent logical structures called 'FDLs'. An internal schema must be defined for each FDL[5].

Given a physical size of a file for an FDL, the module INTERNAL generates a job deck for the definition of an internal schema. All the other parameters like a read/write permission are substituted for dummy values, which can be updated after finishing the database structuring.

The module CONCEPTUAL assists an administrator to edit the logical structure of an FDL. Each line of an FDL is grammatically checked immediately after its input on a TSS terminal. Furthermore, CONCEPTUAL provides another optional mode which translates a DDL of COOD into an FDL of INQ in a certain standardized rule discussed in Sec. 4.

The module EXTERNAL generates a job deck for the definition of an external schema that is defined in a hierarchical connection of FDLs. In DBGUIDE, a DBA only has to type names of connected FDLs and their connection type, and then an external schema is semi-automatically generated according to a DBA's command.

3.4 Data Storage: STORAGE

INQ provides a utility program LOADER which loads a database with mass data and generates simultaniously inverted files for keyword retrieval. Then the module STORAGE generates automatically a job deck for LOADER according to a description of an FDL. Even though the automatic generation is available only for INQ, it may be one of the most helpful functions for untrained DBAs.

3.5 Data Conversion Program: CONVERSN

Before loading a database with mass data, a DBA has to convert the format of source data into the one demanded by LOADER. By analyzing an FDL, the module CONVERSN generates part of a FORTRAN data conversion program in which source data are ignored. Especially for a COOD database, CONVERSN provides an optional mode that completes a data conversion program from COOD to INQ. These algorithms will be discussed in Sec. 5.

3.6 Summary

After finishing the database structuring, a database user is able to have immediate access to the database through an end user language of INQ.

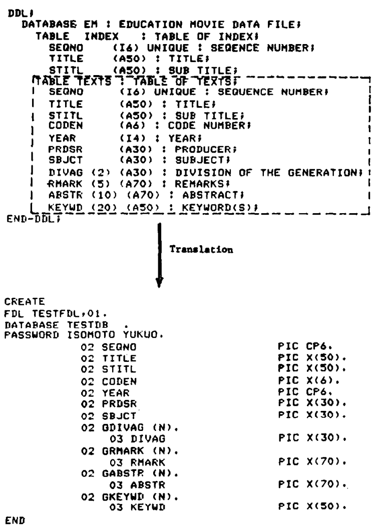
4. Automatic Schema Translation

When a DBA has a source database in a DBMS like COOD, he will need an automatic translation of its logical schema into a target one for his database restructuring. In order to implement an understandable guidance system for restructuring of a database, the module CONCEPTUAL supports an automatic schema translation from COOD into INQ in association with changing its database management strategy: COOD, which is a flat model, is suitable for personal handy manipulative databases. INQ, which is a hierarchical model, is suitable for large scale information retrieval oriented databases.

Surveying both the data description languages of COOD and INQ, the translation rules are formalized

Table 3 Translation rules from COOD to INQ. This table realizes a unique automatic translation of a schema from a COOD database into an INQ one.

	COOD	INQ
Integer	In	CPb; 4-byte binary
Real	Fm.n, Em.n Dm.n, Jm	FB; 4-byte floating number FBD; 8-byte floating number
Character	Am, Rm	X(m)
item	data item array name	data item repeating group



```
DDL;
  DATABASE EM : EDUCATION MOVIE DATA FILE;
    TABLE  INDEX   : TABLE OF INDEX;
      SEQNO     (I6) UNIQUE : SEQENCE NUMBER;
      TITLE     (A50) : TITLE;
      STITL     (A50) : SUB TITLE;
    TABLE TEXTS : TABLE OF TEXTS;
      SEQNO      (I6) UNIQUE : SEQUENCE NUMBER;
      TITLE      (A50) : TITLE;
      STITL      (A50) : SUB TITLE;
      CODEN      (A6) : CODE NUMBER;
      YEAR       (I4) : YEAR;
      PRDSR      (A30) : PRODUCER;
      SBJCT      (A30) : SUBJECT;
      DIVAG (2) (A30) : DIVISION OF THE GENERATION;
      RMARK (5) (A70) : REMARKS;
      ABSTR (10) (A70) : ABSTRACT;
      KEYWD (20) (A50) : KEYWORD(S);
END-DDL;
```

```
CREATE
FDL TESTFDL,01.
DATABASE TESTDB  .
PASSWORD ISOMOTO YUKUO.
          02 SEQNO                  PIC CP6.
          02 TITLE                  PIC X(50).
          02 STITL                  PIC X(50).
          02 CODEN                  PIC X(6).
          02 YEAR                   PIC CP6.
          02 PRDSR                  PIC X(30).
          02 SBJCT                  PIC X(30).
          02 GDIVAG (N).
             03 DIVAG               PIC X(30).
          02 GRMARK (N).
             03 RMARK               PIC X(70).
          02 GABSTR (N).
             03 ABSTR               PIC X(70).
          02 GKEYWD (N).
             03 KEYWD               PIC X(50).
END
```

Fig. 3 An example of automatic translation of a conceptual schema from COOD into INQ. The DDL of COOD was automatically translated into the FDL of INQ by the module CONCEPTUAL. In this figure, JCLs are omitted.

in Table 3. As the logical structure of INQ is much more flexible (hierarchical) than the one of COOD (flat), a COOD database can be translated uniquely into an INQ one in accordance with the rules in Table 3. Because of the flat structure of COOD, these rules are much simpler than the ones in Refs. 13 and 14.

Fig. 3 shows an example of the automatic translation. A DDL of COOD is composed of tables whose logical structures are described with data names and array names. The array names are translated into repeating groups of an FDL notified by the string '(N)'. Since several FDLs can be connected into a single external schema, the translation will be able to realize various access paths by devising external schemas of INQ[5].

5. Analysis of a Schema and Generation of a Data Conversion Program

Generally speaking, database restructuring involves data conversion from a source database into a target one in addition to the schema translation. This section defines notation and show an algorithm for generating a data conversion program from COOD to INQ. For coding a data conversion program as automatically as possible, DBGUIDE analyzes both conceptual schemas of a target INQ database and a source COOD one. Owing to the flat structure of COOD, the data conversion is simpler than the discussions in Refs. 15 and 16.

The module CONVERSN analyzes an FDL according to the following three characteristics: The logical structure of an FDL is hierarchical such as a file description of COBOL. The level number of a repeating group (a member record in ANSI terminology) must be one larger than its higher one (an owner record in ANSI terminology[17]). Due to the flat structure of COOD, a DDL can be analyzed in the same method as an FDL.

For the formal analysis of an FDL (or a DDL), the notation for an i-th item D_i is defined as follows;

$$\begin{aligned} &D_i=\{L_i, N_i, A_i\}, i=1, 2, 3, \cdots, I, \\ &\delta_i=L_{i-1}-L_i \text{ for a repeating group item } D_i, \quad (5.1)\\ &\quad =0 \qquad \text{for else,} \end{aligned}$$

where

L_i:	a level number of D_i,
N_i:	an item name of D_i,
A_i:	an attribute of D_i (a data item, a repeating group, a data length, a data type, et al.),
δ_i:	the difference between level numbers of D_{i-1} and D_i, D_i is a repeating group item and D_{i-1} is a data item.

A line number i is serially counted from the top ($i=1$) to the bottom ($i=I$). CONVERSN determines D_i and δ_i as reading sequentially lines of an FDL from the top to the bottom.

For a simple expression of the generating process of a data conversion program, a set G_k of data items in a k-th repeating group is expressed as follows:

$$G_k=\{N_i|i_{\min}<i<i_{\max}\}, \qquad (5.2)$$

where $D_{i\min}$ is a k-th repeating group item, and $D_{i\max}$ is a $(k+1)$-th repeating group item. The number k is called a record set number that numbers serially individual repeating groups.

Moreover, we introduce an operator [ ] which converts an integer n or a data item name N_i into a character string: For example, if n is an integer 123, [123] is a character string '123'. [N_i] is the character string corresponding to a data item name N_i.

The notations are used to formulate the generation of a job deck of a data loading and a data conversion program. Fig. 4 shows an example of an FDL and its related parameters. According to this formulation, the parameter δ_i predicates the hierarchical structure of an FDL: When $\delta_i=0$, $G_k(D_i \in G_k)$ is a member record of an owner G_{k-1}. When $\delta_i=1$, G_k is on the same level as G_{k-1}. Moreover, when $\delta_i \geqq 2$, G_k is on the (δ_i-1) level above G_{k-1}.

i	L_i	N_i	A_i	δ_i	G_k
1	01	DATABASE-NAME.			G_1
2	02	PRIMARY-KEY	PIC X(4).		
3	02	DATA-A	PIC X(4).		
4	02	GROUP-B	(N).	0	G_2
5	03	DATA-B1	PIC FB.		
6	03	DATA-B2	PIC CP6.		
7	03	GROUP-C	(N).	0	G_3
8	04	DATA-C	PIC X(10).		
9	03	GROUP-D	(N).	1	G_4
10	04	DATA-D	PIC X(10).		
11	02	GROUP-E	(N).	2	G_5
12	03	DATA-E	PIC FB.		
13	03	GROUP-F	(N).	0	G_6
14	04	DATA-F	PIC X(50).		
15	03	GROUP-G	(N).	1	G_7
16	04	DATA-G	PIC CP6.		
17	04	GROUP-H	(N).	0	G_8
18	05	DATA-H	PIC FB.		
19	02	GROUP-I	(N).	3	G_9
20	03	DATA-I	PIC X(4).		
21	03	GROUP-J	(N).	0	G_{10}
22	04	DATA-J	PIC FB.		
23	02	END	.		

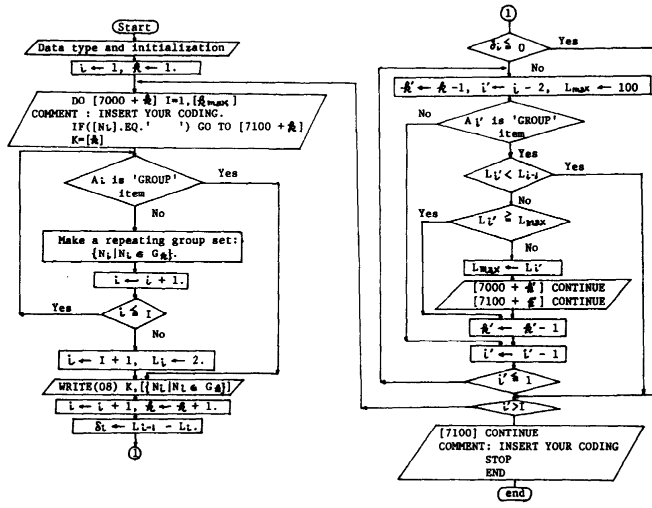
Fig. 4 An example of an FDL and its associative parameters. The parameters are defined in Eqs. (5.1) and (5.2).

Fig. 5 shows a flow diagram for the automatic generation of a FORTRAN data conversion program, in which data transference from a source database is left incomplete to be coded later. For a source COOD database, comment statements in the diagram are exchanged for data manipulation languages of COOD (DMLs).

Fig. 6 shows a data conversion program corresponding to the shema translation in Fig. 3. The program was generated through man-machine interactions in Fig. 2. This program has six DMLs of COOD,. USE, OPEN, .FIND, .IF, .GET, and .CLOSE, which are later translated into ordinary FORTRAN statements by the COOD precompiler. The pair combination of .FIND and .GET transfers a data set to a program working area at one time. Array names such as DIVAG, RMARK, ABSTR, and KEYWD are written in DO loops. IREC corresponds to a record set number ($K=[k]$ in Fig. 5) of a repeating group.

```
00010#$,J N
00020$      JOB     6000820405$[redacted],B
00030$      LIMITS  25,,,6000
00040$      LOWLOAD
00050$      OPTION  FORTRAN,RELMEM
00060$      FORTRAN LSTIN,NLNO,NFORM
00070CCC  FORTRAN PROGRAM FOR DATA CONVERSION  CCC
00080CCC  CONNECT COOD SCHEMA TO INQ SCHEMA  CCC
00090      .USE EM/TEXTS(ALL-ITEMS);
00100      .OPEN TEXTS;
00110C*** UNIQUE ITEMS ***
00120      DO 7001 I01=1,  1200
00130      .FIND TEXTS;
00140      .IF END(TEXTS),GO TO 8000;
00150      .GET SEQNO,TITLE,STITL,CODEN,YEAR,PRDSR,SBJCT,DIVAG,RMARK,ABSTR,KE
00160     &YWD
00170      IF(SEQNO.EQ.000000) GO TO 8000
00180      IREC=01
00190      WRITE(08) IREC,SEQNO,TITLE,STITL,CODEN,YEAR,PRDSR,SBJCT
00200C*** GROUP DNAME IS GDIVAG ***
00210      DO 7002 I02=1,     2
00220      IF(DIVAG(I02).EQ.'    ') GO TO 7102
00230      IREC=02
00240      WRITE(08) IREC,DIVAG(I02)
00250 7002 CONTINUE
00260 7102 CONTINUE
00270C*** GROUP DNAME IS GRMARK ***
00280      DO 7003 I03=1,     5
00290      IF(RMARK(I03).EQ.'    ') GO TO 7103
00300      IREC=03
00310      WRITE(08) IREC,RMARK(I03)
00320 7003 CONTINUE
00330 7103 CONTINUE
00340C*** GROUP DNAME IS GABSTR ***
00350      DO 7004 I04=1,    10
00360      IF(ABSTR(I04).EQ.'    ') GO TO 7104
00370      IREC=04
00380      WRITE(08) IREC,ABSTR(I04)
00390 7004 CONTINUE
00400 7104 CONTINUE
00410C*** GROUP DNAME IS GKEYWD ***
00420      DO 7005 I05=1,    20
00430      IF(KEYWD(I05).EQ.'    ') GO TO 7105
00440      IREC=05
00450      WRITE(08) IREC,KEYWD(I05)
00460 7005 CONTINUE
00470 7105 CONTINUE
00480 7001 CONTINUE
00490 8000 .CLOSE TEXTS;
00500      STOP
00510      END
00520$      LIBRARY CL
00530$      EXECUTE
00540$      PRMFL   CL,R,R,LIB/COOD/DME
00550$      FILE    08,A1S,100L
00560$      LIMITS  6,50K,-4K,6000
```

Fig. 6 A job deck for a data conversion program. The job deck was automatically generated by the module CONVERSN according to the flow diagram in Fig. 5. The program converts data from a COOD database into an INQ one in consistency with the translation of Fig. 3.



Start
Data type and initialization
$i \leftarrow 1,\ k \leftarrow 1.$
DO [7000 + k] I=1, [k_{max}]
COMMENT : INSERT YOUR CODING.
IF([N_i].EQ.'   ') GO TO [7100 + k]
K=[k]
A_i is 'GROUP' item
Yes
No
Make a repeating group set: $\{N_i \mid N_i \in G_k\}$.
$i \leftarrow i + 1.$
Yes
$i \leq I$
No
$i \leftarrow I + 1,\ L_i \leftarrow 2.$
WRITE(08) K,[$\{N_i \mid N_i \in G_k\}$]
$i \leftarrow i + 1,\ k \leftarrow k + 1.$
$\delta_i \leftarrow L_{i-1} - L_i.$
①
①
$\delta_i \leq 0$
Yes
No
$k' \leftarrow k - 1,\ i' \leftarrow i - 2,\ L_{max} \leftarrow 100$
No
$A_{i'}$ is 'GROUP' item
Yes
$L_{i'} < L_{i-1}$
Yes
No
Yes
$L_{i'} \geq L_{max}$
No
$L_{max} \leftarrow L_{i'}$
[7000 + k'] CONTINUE
[7100 + k'] CONTINUE
$k' \leftarrow k' - 1$
$i' \leftarrow i' - 1$
$i' \leq 1$
$i' > I$
[7100] CONTINUE
COMMENT: INSERT YOUR CODING
STOP
END
end

Fig. 5 Flow diagram for the generation of a data conversion program. The notations are defined in sect. 5. The flow diagram is applied to a case in which source data are not definite. When source data are stored in a COOD database, DBGUIDE exchanges "COMMENT: INSERT YOUR CODING." with the verbs of COOD as follows:

·FIND, ·GET, and ·IF for $i=1$.
·CLOSE for the last one,
others are removed.

The data conversion can be classified into three levels:

Level 1 A COOD database is automatically converted to an INQ database in the formal procedure.

Level 2 A part of a conversion program is generated in consideration of an FDL. In this level, the format of source data is disregarded, and a program is left incomplete to be coded by a DBA.

Level 3 For a more complex data conversion, a data conversion program is coded by a DBA. This level is beyond the scope of this paper.

A DBA can select one of the levels corresponding to the difference between both the source data and an INQ database.

6. Discussion and Conclusion

Up to the present, a DBMS has been discussed almost from a professional point of view. On the one hand, this paper discusses the database structuring guidance for non-professional DBAs. A guidance system like DBGUIDE plays a very important role in familiarizing untrained DBAs with the use of a DBMS. By standardizing the database structuring on INQ, DBGUIDE is designed as not only a guidance system but also a learning tool for untrained DBAs. When a DBA becomes an expert with the assistance of DBGUIDE, he will be able to update his database by himself.

This experimental system could not only manifest an understandable systematic guidance of the database structuring but also implement a practical system by separating the standardized manipulation of a DBMS from the operations of job decks on a host computer. DBGUIDE is just an elementary prototype for future improved systems, it will be revised to satisfy more general requirements in database structuring.

Acknowledgements

The authors express their sincere gratitude to all members of the Research Project "Information Studies on Trace Characterization" and also thank the Ministry of Education, Science, and Culture for financial support (Project nos. 411701, 510801, and 56104001). The authors wish also to express their thanks to NEC for the maintenance of DBGUIDE during its users' service.

References

1. Data for Science and Technology, Proceedings of the seventh International CODATA Conference, Kyoto, Japan (October 1980) 8–11.
2. TANAKA, N. et al. Computer databases and their applications in analytical chemistry: I. Database on complexation reactions, *BUNSEKI KAGAKU*, **30**, 9 (1981), 588–594.
3. TANAKA, N. et al. Preliminary study of a trace characterization information system, *Anal. Chem. Acta/CTO*, **133**, 4 (1981).
4. ISOMOTO, Y. et al. Protein Database for Scientific Researchers, PROTEIN-DB, *Transaction of Information Processing Society of Japan*, **21**, 1 (1980), 15–22.
5. HASHIMOTO, M. et al. Database Management System; INQ (Information Query), *NEC Research & Development*, *NEC*, **58** (1980), 33–41.
6. MATSUDA, T. et al. User-oriented Database Management System COOD—*Its design and database languages*, **21**, 5 (1980), 347–353.
7. TANAKA, N. et al. User-oriented database management system, COOD, and its applications to on-line data storage and retrieval, *Data for Science & Technology. Pergamon* (1981), 504.
8. ROTHNIE, J. B. et al. Introduction to a System for Distributed Database (SDD-1), *ACM Trans. Database Syst.*, **5** 1 (March 1980), 1–17.
9. MCLEOD, D. and HEIMBIGNER, D. A federated architecture for database systems, *National Computer Conference* (1980), 283–289.
10. SMITH, J. M. et al. Multibase-integrating heterogeneous distributed database systems, *National Computer Conference* (1981), 487–499.
11. SHNEIDERMAN, B. Improving the Human Factors Aspect of Database interactions, *ACM Trans. Syst.*, **3**, 4 (December 1978), 417–439.
12. KERSTEN, M. L. et al. The Architecture of the PLAIN Database Handler, *Software-Parctice and Experience*, **11** (1981), 175–186.
13. NAVATHE, S. B. et al. Restructuring for Large Databases: Three Levels of Abstraction, *ACM Trans. Database Syst.*, **1**, 1 (June 1976), 138–158.
14. FRY, J. P. et al. An assessment of the technology for data- and program-related conversion, *National Computer Conference* (1978), 887–907.
15. NAVATHE, S. B. Schema Analysis for Database Restructuring, *ACM Trans. Database Syst.*, **5**, 2 (June 1980), 157–184.
16. SU, S. Y. W. et al. Transformation of Data Traversals and Operations in Application Programs to Account for Semantic Changes of Databases, *ACM Trans. Database Syst.*, **6** 2 (June 1981), 255–294.
17. JARDINE, D. A. THE ANSI/SPARC DBMS MODEL, North-Holland Publish Company (1976).

(Revised January 5, 1982)