THE DESIGN OF AN INTEGRATED DATA DICTIONARY DIRECTORY SYSTEM

by

Ryosuke Hotaka
The University of Tsukuba
Ibaraki, 305, Japan

ABSTRACT

As the processes of information system development and maintenance became complex, the needs for controlling and maintaing the processes themselves came out. Data Dictionary/Directory (DD/D) was developed after the experience of the database management system to complement the above needs.

Owing to its historical constraints, DD/D had to use the then available tools such as conventional files or databases. It turned out, however, that one cannot overcome some intrinstic deficiency so far as relying on existing tools. In particular, the departure of DD/D from DBMS causes theoretically unfavorable problems.

The integrated DD/D (GUIDE (1974)) overcomes these problems. The practical architecture of an integrated DD/D using the self-descriptive DBMS (Hotaka and Tsubaki (1977)) is described with a concrete example.

Some efficiency degradation, perhaps intrinsic in the integrated DD/D and DBMS, is observed and the practical method of optimization is stated.

1. INTRODUCTION

As the process of information system development and maintenance became complex, the needs for controlling and maintaining the processes themselves came out. Data Dictionary/Directory (DD/D) was developed after the experience of the data base management systems to complement the above needs.

Owing to its historical constraints, DD/D had to use the then available tools such as conventional files or databases. It turned out, however, that one cannot overcome some intrinsic deficiency so far as relying on existing tools. In particular, the departure of DD/D from DBMS causes theoretically unfavorable problems.

The integrated DD/D (GUIDE (1974)) overcomes these problems, but so far, the practical way of implementation is not well known. The author has proposed one method (Hotaka and Tsubaki (1977)) using self-descriptive DBMS. Here the method is restated emphasizing the DD/D aspect.

In 2, various architecture and applications of DD/D are briefly reviewed. Deficiencies of non-integrated DD/D are pointed out in 3. New approach using self-descriptivity, is introduced to realize an integrated DD/D in 4. DD/D is the meta database or the meta information and is itself a kind of database. Its schema is defined in 5 and an example of the occurrences of DD/D is given in 6. In 7, the use of DD/D is described from two points of views. One is the use of DD/D by the DBMS and the other by the application programs.

DD/D is indispensable and plays important role in installing distributed databases. But only non-distributed database environment will be considered in this paper. Relations between DD/D and distributed databases are stated elsewhere (Tsubaki and Hotaka (1979)).

2. REVIEW OF THE DD/D SYSTEMS

Many authors state the merits amd necessity, the objects of descriprion, basic functions and application of DD/D. We will briefly review these points.

MERITS AND NECESSITY

Not all data of enterprises have been stored in computer much less in databases. Therefore all the problems of data cannot be solved using only data base management systems, hence the necessity of DD/D (Leon-Hong and Marron (1977)). Some of these problems are (Lee and Lee (1975)):

- Data redundancy
- Lack of standardization
- Lack of sources or derived information of the data
- Lack of comprehensive description of data
- Lack of tools to make estimates of the effort of changing the database
- Lack of centralized tools for the DBA to effectively carry out his/her duties.

Whereas the merits come from DD/D are (Leong-Hong and Marron (1977))

- Simple and effective control of the data elements
- Reduction of data redundancy and inconsistency
- Enforcement of standard usage
- Enforcement of security safeguards and controlled accessibility to the database
- Determination of the impact on the total information activity from changes to data elements
- Centralization of data elements as an aid in design and development of new systems
- Consistency in documentation for data elements

THE OBJECTS OF DESCRIPTION

DD/D must describe all the information of an installation that needs management. There are many ways of constructing DD/D's, but roughly speaking, they are divided into three categories of information: data, processes, user/owner.

There may be three kinds of data: data related to the database such as

- . Databases
- . Database files
- . Database records
- . Database group of data items
- . Database data items

and data related to the conventional files such as

- . Files
- . Records
- . Group of data items
- . Data items

and data related to non-mechanized information such as

- . Manual files
- . Documents
- . Transactions.

There may be two kinds of processes: mechanized processes such as

- . Procedures (an ordered set of programs)
- . Programs
- . Modules

and non-mechanized processes such as

- . Systems (a composite of subsystems)
- . Subsystems (a composite of procedures and manual procedures)
- . Manual procedures.

Non-Mechanized procedures are playing important roles at the present state of information processing. Therefore they need careful management by DD/D.

There may be two kinds of user/owners.

- . Organizations (such as department, division and so forth)
- . Persons

Relationships among these objects are also described.

BASIC FUNCTIONS

Basic functions usually provided by the DD/D are as follows (Leong-Hong and Marron (1977), DDSWP (1977)).

- . Various reporting facilities such as cross-reference reports, change effect reports, error-reports etc.
- . Various retrieval capabilities such as keywording, indexing, and online or batch querying.

. Command language to control, retrieve and update the DD/D.
. Validation and redundancy-checking capabilities
. Security safeguards to control the accessibility to the DD/D
. Data description generation

APPLICATIONS OF DD/D

DD/D can be used in various situations of information systems using the above-stated basic functions. Here we observe important application of DD/D, especially in development and production of data processing systems (Canning (1978), DDSWP (1977), Leong and Marron (1977), Lyon (1976), Towner (1977)).

In development phase:

. A designer may ask DD/D all the documents relating to the program in which he is now engaged.
. Suppose a designer needs a data item in the database. He wants to know if the data items has already been defined in the data base.
. In integrating two files into a database, the DBA must determine if the two similar data items in each files are identical. The process of reconciliation is facilitated if the DBA can identify the source of data, its meaning and where and how it has been maintained.
. DBA must ascertain if there are any data of which source have not been defined.
. He also must find out the data which will be used by a program or a procedure and has not been defined yet.
. In creating data, the DBA checks if the name obeys the standard naming rules described in DD/D.
. Suppose a change in the meaning or representation of data occurrs. The DBA must analyze the impact of that change using the relationship between the data and the other resources described in DD/D.

In production phase:

. DD/D can monitor the usages of data or programs. Therefore DBA can find these resources which are not used. Frequently one of redundant data items can be discovered this way.
. He can also know the frequency of usages of the resources to tune up the system.
. Sometimes the replicated data are created for the efficiency. The replication is described in and controlled by the DD/D.
. DD/D generates data definitions for DBMS and ordinary programming languages.
. DD/D provides useful information for users. E.g. it is useful for determining what resources are represented in the database, what limitations are imposed, and what procedures or systems may have already dealt with a particular kind of the resources.

There are not only advantages but also disadvantages to the DD/D (Leong and Marron (1977)). DD/D can be time-consuming to install; the maintenance function may require considerable effort; and there may be objection to the formality necessitated by the DD/D.

ARCHITECTURES OF DD/D's

From architectural point of view, DD/D is classified into two categories:

- integrated DD/D
- non-integrated DD/D

An integrated DD/D is an integral component of a database management system (DBMS). As a result, only a single set of descriptions is required (GUIDE (1974)).

There is another way of classification of DD/D:

- dependent DD/D
- independent DD/D

A dependent DD/D uses a particular DBMS to store its description, wherease independent DD/D uses conventional access methods, e.g. ordinary data management facilities of an operating system. Thus an integrated DD/D is necessarily a dependent DD/D.

At present, all of the commercially available DD/D's are non-integrated. DATA CATALOGUE (Synergetics Corp.), DATA MANAGER (MSP, Inc.), LEXICON (Arthur Andersen & Co.), PRIDE/LOGIK (M.Bryce & Assoc.) are independent DD/D's. DB/DC Dictionary (IBM), TIS Directory (Cincom Systems, Inc.), UCC Ten (University Computing Co.) are dependent DD/D's.

Dependent DD/D's may seem deficient in that it has close relationship with a DBMS and has no ability to adapt to other DBMS's. But this is not so because one can implement various interfaces to many DBMS's as they do in independent DD/D's.

3. DEFICIENCIES OF NON-INTEGRATED DD/D

Both integrated and non-integrated DD/D have their merits and demerits. But, for theoretical completeness, integrated DD/D is superior to non-integrated one (GUIDE (1974)). We will advocate the integrated DD/D, and observe deficiencies of non-integrated DD/D or superiority of integrated DD/D.

The utmost cause of deficiency is the differences between two descriptions: the one held by the non-integrated DD/D and the other held by the DBMS. The DBA must always be cautious so that the description in the DD/D should not be taken seriously unless the corresponding description is in the DBMS. For example, if he register a new data item in the DD/D, he must attach status flag to that data item to show whether it is real (i.e. DBMS is in the same status with respect to the data item) or not (i.e. the description is only local to the DD/D and has no relation with the real activity of the DBMS). And when the flag showing that the data item is real is set,

the DBA must control the DBMS so as no users access the database before the same description is memorized by the DBMS. The DBMS is particularly useful in avoiding redundant data thus alleviating the burden of the DBA. Non-integrated DD/D, though it is a kind of a database itself, cannot eliminate the redundancy of the description, thus increasing the load of the DBA.

Another deficiency is the problem of consistency. Suppose certain kinds of security of integrity constraints are imposed on some data, e.g. a data item containing the payroll should be protected from accesses by persons belonging to the other department than the department of personnel.

Even if the restriction of this kind is stated in the DD/D, it has no influence on the actual execution by the DBMS since the DBMS simply dose not know the existence of the DD/D at all. In non-integrated DD/D situations, DD/D is completely exterior to the DBMS and it is only a kind of application of the data management facility of the operating system.

The DBA would have to have checked the logic of all the application programs before execution if he is to maintain the security or integrity conditions as stated in the DD/D. In reality, it would be almost unable for him to preserve security or integrity unless all of the application programs are developed by a team of programmers where these security or integrity conditions are enforced by other, say administrative or managerial, methods.

In the integrated DD/D, these problems all disappear. The DBA can impose any integrity or security conditions supported by the DBMS simply by adding or inserting the condition in the DD/D. Here, the description in the DD/D implies the real control of the data, though the control is limited to the extent that the DBMS supports.

4. AN APPROACH TO THE INTEGRATED DD/D

An integrated DD/D and its associated DBMS DPLS was implemented (Hotaka and Tsubaki (1977)). Its Architectural problems are restated focusing the discussions on the integrated DD/D. The DD/D is itself a part of the database, and the corresponding schema will be explained in 5.

The definition of the entire database including its control information, the so called data directory, is itself described using the same method which the database uses for describing ordinary data. Since the DD/D is a part of the database, it is described by the same method. Thus it is described self-descriptive. An integrated DD/D could be built without using self-descriptivity if the DBMS uses the data directory in the DD/D as its control information. But the description of the control information itself would not be described in the DD/D, thus the risk of discrepancies of this kind of information. Theoretically, self-descriptive construction gives the complete integrated DD/D.

Curiously enough, in a self-descriptive database, a database occurrence (supposing usual "data type" and "data occurrence" terminology also applies to database) explains the schema of the database. A database occurrence example will be explained in 6.

A DBMS with any database model could be built self-descriptive, but the model using flat files or tables is adopted in this paper.

5. THE SCHEMA OF THE DD/D

The real DD/D contains all the information from logical data description to the detailed physical data description. It also varies according to the model of the information system and admits various extentions added by particular installations. We will assume flat tables or the relational model and simplify the description for the ease of understanding.

The schema diagram (in Bachman notation) of Fig.1 shows how the tables or relations are related to each other. The exact schema is described by specifying relation names and accompanying attributes.

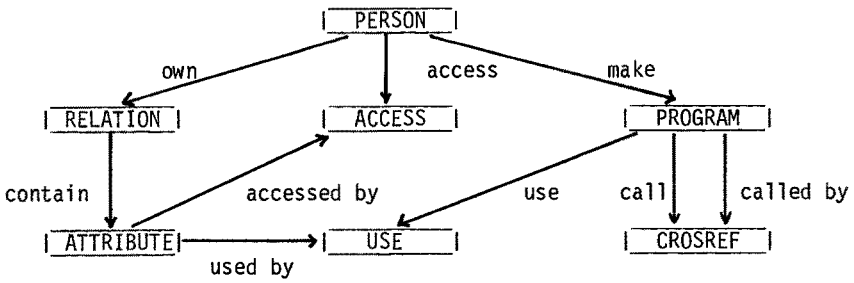

Fig.1 The schema diagram of the DD/D

THE SCHEMA

Each relation must have an attribute which contains the tuple identifier. We first describe the schema of the relation RELATION.

RELATION (RELID, RNAM, OWNER, LOC, TLEN, TIDATRNO)
RELID : the tuple identifier
RNAM : the name of the owner of the relation
OWNER : the name of the owner of the relation
LOC : the physical location where the relation is stored
TLEN : the length of a tuple
SIZE : the size of the space allowed to the relation
TIDATRNO: the tuple id of the attribute which plays the role of tuple identifier in the relation, e.g. the TIDATRNO of the relations RELATION, ATTRIBUTE, PERSON are 1, 11, 21 respectively.

ATTRIBUTE (ATRID, ANAM, REL, DTYPE, LEN, OFFSET)
ATRID : the tuple identifier
ANAM : the name of an attribute
REL : the relation identifier of the relation to which the attribute belongs
DTYPE : the data type of the attribute, e.g. AN(alphanumeric), N(numeric)
LEN : the length of the attribute
OFFSET: the relative position of the attribute in each tuple occurrence

PERSON (PID, PNAM, DEPT)
PID : the tuple identifier
PNAM : the name of a person
DEPT : the department to which the person belongs

PROGRAM (PGMID, PGMNAM, AUTHOR)
PGMID : the tuple identifier
PGMNAM: the name of a program
AUTHOR: the name of the author of the program

ACCESS (ACCID, ACATR, UNAM, ACOND)
ACCID : the tuple identifire
ACATR : the name of an attribute accessible to the user
UNAM : the name of a person having the right to access the attribute
ACOND : the access condition, e.g. R(read only), W(read and write)

USE (USEID, UATR, UPGM)
USEID: the tuple identifier
UATR : the name of an attribute used by a program
UPGM : the name of the program which uses the attribute

CROSREF (CROSID, MPGM, SPGM)
CROSID: the tuple identifier
MPGM : the name of the main program which calls a subprogram
SPGM : the name of the subprogram which is called by the main program

The schema of the DD/D could be designed as complex as the system requires, but only the simplest case is illustrated here. For example, all the length of attributes are assumed to be fixed, information concerning to non-database data are omitted, optimization in constructing relations are not tried.

6. AN EXAMPLE OF THE DD/D OCCURRENCE

Occurrences of two relations RELATION and ATTRIBUTE are exhibited as an example (Fig.2 and Fig.3). They constitute the kernel of the integrated DD/D. Note the ordinary files or relations in the database will be handled analogously.

In the relation RELATION, the first tuple carries the information of the relation RELATION itself, i.e. the RELATION is described in the RELATION, hence this representation is considered to be self-descriptive. In the same way, the relation

ATTRIBUTE, contains the tuples relating to the attributes ATRID, ANAM, REL, DTYPE, LEN, OFFSET which again are used to describe the contents of the relation ATTRIBUTE itself.

RELATION relation

RELID	RNAM	OWNER	LOC	TLEN	NOOF TIDS	TID ATR NO
1	RELATION	DBA	0	42	500	1
2	ATTRIBUTE	DBA	500	24	1000	11
3	PERSON	PRESIDENT	1500	28	100	21
4	PROGRAM	DBA	1600	28	400	31
5	ACCESS	DBA	2000	29	200	41
6	USE	DBA	2200	28	100	51
7	CROSREF	DBA	2300	28	200	61

Fig. 2 The relation RELATION which describes all the relations in the database

7. THE USE OF THE DD/D

HOW THE SYSTEM USES THE DD/D

Any complex database operations can be decomposed into a small group of elementary operations such as

Add(r ; t):
Newly add a tuple in the specified relation the relation identifier (the tid of the tuple which corresponds to the relation in the relation RELATION) of which is equal to r. Then the tuple identifier of the new tuple is assigned to t.

Delete(r, t):
Delete the tuple in the relation the relation identifier of which is equal to r and the tuple identifier of the tuple is equal to t.

Getvalue(a, t; v):
Get the attribute value the attribute identifier (the tid of the tuple which corresponds to the attribute in the relation ATTRIBUTE) of which is equal to a and the tuple identifier is equal to t. Then the value is assigned to v. Note, in our model, it is not necessary to specify any relation because it is determined by the attribute.

E.g. if RNAM=2 and OWNER=3 then Getvalue(RNAM, 1; v) –> v = 'RELATION', Getvalue(OWNER, 1; v) –> v = 'DBA', Getvalue(RNAM, 3; v) –> v = 'PERSON', Getvalue(OWNER, 3; v) –> v = 'PRESIDENT'. See also the section "HOW THE USERS USE THE DD/D" below about the relation between Getvalue and the primitive commands such as Getatr or Getrel.

ATTRIBUTE relation

ATRID	ANAM	REL	DTYPE	LEN	OFFSET
1	RELID	1	N	4	0
2	RNAM	1	AN	12	4
3	OWNER	1	AN	12	16
4	LOC	1	N	4	28
5	TLEN	1	N	2	32
6	NOOFTIDS	1	N	4	34
7	TIDATRNO	1	N	4	38
11	ATRID	2	N	4	0
12	ANAM	2	AN	12	4
13	REL	2	N	2	16
14	DTYPE	2	AN	2	18
15	LEN	2	N	2	20
16	OFFSET	2	N	2	22
21	PID	3	N	4	0
22	PNAM	3	AN	12	4
23	DEPT	3	AN	12	16
31	PGMID	4	N	4	0
32	PGMNAM	4	AN	12	4
33	AUTHOR	4	AN	12	16
41	ACCID	5	N	4	0
42	ACATR	5	AN	12	4
43	UNAM	5	AN	12	16
44	ACOND	5	AN	1	28
51	USEID	6	N	4	0
52	UATR	6	AN	12	4
53	UPGM	6	AN	12	16
61	CROSID	7	N	4	0
62	MPGM	7	AN	12	4
63	SPGM	7	AN	12	16

Fig. 3 The relation ATTRIBUTE which describes all the attributes in the database

Putvalue(a, t, v):

Put the value v into the attribute value of the tuple the tuple identifier of which is equal to t and the attribute is equal to a.

These elementary operations are further decompoesd into more fundamental primitive commands such as

Getatr(ma, a; v):
Suppose the tuple identifier of the attribute be a and the specified (meta-)attribute has the tuple identifier equal to ma. Then the (meta-)attribute value of the tuple is assigned to v.

E.g. Getatr(12, 1; v) -> v = 'RELID', Getatr(13, 1; v) -> v = 1,
Getatr(14, 1; v) -> v = 'N', Getatr(15, 1; v) -> v = 4,
Getatr(16, 1; v) -> v = 0, Getatr(11, 13; v) -> v = 13, Getatr(12, 12; v) -> v = 'ANAM', Getatr(13, 13; v) -> v = 2, Getatr(14, 14; v) -> v = 'AN'.

Getrel(ma, r; v):
Suppose the tuple identifier of the relation be r and the specified (meta-)attribute has the tuple identifier equal to ma. Then the (meta-)attribute value of the tuple is assigned the value v.

E.g. Getrel(2, 2; v) -> v = 'ATTRIBUTE', Getrel(3, 2; v) -> v = 'DBA'.

and the physical input/output commands such as

DB read(loc, len; v):
The data of length len beginning from the physical location loc in data storage is read and assigned to v.

DB write(loc, len, v):
The data v of length len is written into the data storage beginning from the physical location loc.

Using these primitive commands, the above stated elementary operations can be defined. The definition is written by the help of ALGOL-like language. For example, Getvalue operation is defined as follows.

```
SUBROUTINE Getvalue(a, t; v)
begin
integer loc, tlen, offset, atrlen, ws;
integer LOC, TLEN, OFFSET, LEN, REL;
     LOC:= 4; TLEN:= 5; OFFSET:= 16; LEN:= 15; REL:= 13;
     Getatr(REL, a; ws); comment get the relation id;
     Getrel(LOC, ws; loc); comment get the beginning location;
     Getrel(TLEN, ws; tlen); comment get the tuple length;
     Getatr(OFFSET, a; offset); comment get the offset of the attribute value;
     Getatr(LEN, a; atrlen); comment get the attribute value length;
     DB read(loc + tlen*(t-1) + offset, atrlen; v); comment see the Fig. 4;
end
```

Other operations could be defined analogously using primitive comments.

Optimization

When an input/output operation is issued by an application program, the operation is decomposed into several elementary operations. And each of the decomposed operation is executed as a sequence of primitive commands as we saw in the above example.

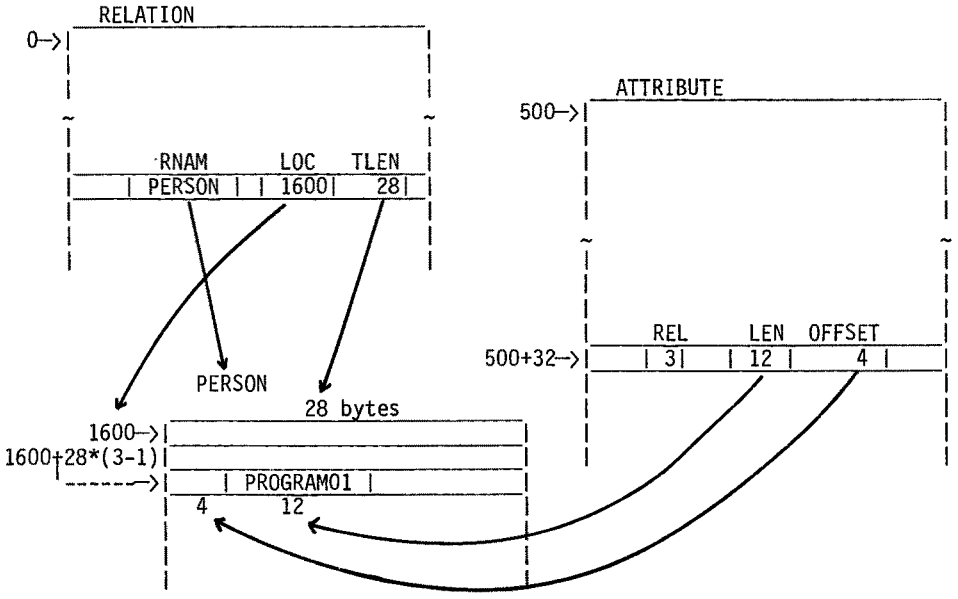

Fig. 4 Getvalue (32, 3; v)

But each primitive command is composed of more detailed instructions. For example, Getatr is described as follows.

```
SUBROUTINE Getatr(ma, a; v)
begin
integer offset, len, loc, tlen;
     loc:= 500; tlen = 24;
     offset := (ma=11)*0 + (ma=12)*4 + (ma=13)*16 + (ma=14)*18 + (ma=15)*20 +
     (ma=16)*22; comment a boolean expression takes value 1 if it is true
     else 0;
     len := (ma=11)*4 + (ma=12)*12 + (ma=13)*2 + (ma=14)*2 + (ma=15)*2 +
     (ma=16)*2;
     DB read(loc + (a-1)*tlen + offset, len; v);
end
```

Note the boolean expression such as (ma=12) is supposed to take value 1 if ma=12 and takes value 0 if ma≠12. Ordinarily physical accesses to the data storage accompanies with DBread or DBwrite commands, hence it may take several times of accesses in one elementary operation such as Getvalue (See the description of the subroutine Getvalue above).

Since the relation ATRRIBUTE or RELATION is not so large compared to ordinary users' relations, it is wise to access these tables in a special way. Possible optimizations are thus

. make the relation ATTRIBUTE or RELATION be main memory resident

provide the special coding to realize the Getatr and Getrel commands without using DBread command.

If other relations are expected to be accessed frequently, then the same special coding for them may be available. But the two most frequently accessed relations are ATTRIBUTE and RELATION.

As the reader might have conceived, Getatr and Getrel subroutines cannot be coded without some apriori knowledge of the kernel relation REL and ATTRIBUTE. Therefore, we cannot change the definition of them separately from the definition of the two kernel subroutines, hence the name of primitives. In this respect, integrated DD/D does not show the complete coincidence with the DD/D and the DBMS if we change the kernel information. But we have shown the utmost range where we can keep the consistency between the DD/D and the DBMS.

In real situation, several of meta-attribute values are necessary at the same time. So it is convenient to get and put values tuple-wise when dealing with kernel relations.

Database generation

The self-descriptive database management system can grow by itself when only small number of kernel subroutines are provided. But in reality, it is less cumbersome to generate the minimum DD/D and get started. This technique can always be applicable when the parent DBMS exists, because he can make the copy of himself. But this cannot be achieved when no parent DBMS exists. The generation or creation of self-descriptive DBMS from nothing is an interesting topic but will be left as a future problem.

HOW THE USERS USE THE DD/D

A DD/D can be used as a powerful database design(Hotaka(1981)) and information administration tool. One can store various useful information in the DD/D and can make queries to it or analyze it to help solve software engineering problems. For example, the impact analysis of the data change explained in 1 can be analyzed by cross-referencing these programs which uses the data. These analysis will be programmed as an ordinary application program.

In the self-descriptive DBMS, a user can retrieve every data of the kernel or system-oriented relations completely the same way as he retrieves the data in ordinary relations. Thus he can make use of the consistent definition of the DBMS. He can, for example, issue the following command

```
Getvalue(15, 32; v).
```

His intention was to retrieve the length of the system attribute PGMNAM. In this case, the command is executed as defined in the SUBROUTINE Getvalue. This logic is rather redundant since we could obtain the same result by issuing

Getatr(15, 32; v).

Of course DBMS allows users to use this kernel commands. But the important thing is that users can get all data in the DD/D by the same command Getvalue. Hence an ordinary query language can interrogate the DD/D as well as the ordinary relations.

The update to the DD/D is more delicate. Sometimes, a user can modify the content of the DD/D by using the same interface. But there does exist the data in the DD/D which cannot or should not be updated (e.g. those duplicated and common information in the kernel relations). And the locking mechanism for these relations requires more delicated controls (Hotaka and Tsubaki (1977)). But again these are left for future problems.

CONCLUSION

The self-descriptive construction and DBMS was proposed as a mechanism to realize an integrated DD/D and some architecture for its implementation was shown. It is said (Lefkovitz (1978)) that the DBMS and the operating system will be integrated in future. In that case, the self-descriptive DD/D will be a good candidate of the architecture since it realizes the complete integrated DD/D. At the present state of the art, self-descriptive architecture is rather slow in execution. But it is expected that future advances in hardware, e.g. an associative memory of several thousand bytes will help much.

ACKNOWLEDGEMENT

The author is very grateful to Dr. Tsubaki of Nihon Systemix Corporation who designed the integrated DBMS DPLS. Mr. H.Sugata of Nihon Sogo System also showed me an elegant recursively structured command system before.

REFERENCES

Arthur Andersen & Co. (1977), LEXICON General Description Manual, Arthur Andersen & Co., 1977.

Cahill,J.J. (1970), A Dictionary/Directory Method for Building a Common MIS Data Base, Journal of Systems Management, Nov. 1970, pp.23-29.

Canning,R. (1974), The Data Dictionary/Directory Function, EDP Analyzer, 1974.

Canning,R. (1978), Installing a Data Dictionary, EDP Analyzer, 16,1, Jan. 1980.

Cullinane Corporation (1980), Database Design and Definition Guide, Revision 1 Release 5.0, Cullinane Corporation, August 1978.

Curtice,R.M. (1974), Some tools for Data Base Development, Datamation, July 1974, pp.102-106.

Data Dictionary Systems Working Party (1977), The British Computer Society, Data Dictionary Systems Working Party Report, SIGMOD Record, 9,4, Dec. 1977.

DDSWP: BCS Computer Society Data Dictionary Systems Working Party (1977), The British Computer Society Data Dictionary Systems Working Party Record, SIGMOD Record, 9, 4, Dec. 1977.

Ehrensberger,M. (1977), Data dictionary - More on the impossible dream, NCC, 1977, pp.9-11.

GUIDE Report for the GUIDE Dictionary/Directory Project (1974), Requirements for the Data Dictionary/Directory Within the GUIDE/SHARE Data Base Management System Concept, GUIDE Data Dictionary/Directory Project, GUIDE (1974).

Hotaka,R. and Tsubaki,M. (1977), Self-descriptive Relational Data Base, Third International Conference on Very Large Data Bases, 1977, pp.415-426.

Hotaka, R. (1981), Logical design of databases, in the monograph series of information processing, Information Processing Society of Japan, forthcoming in 1981.

Lee,E.K.C. and Lee,E.Y.S. (1975), Development of a Data, Dictionary/Directory for Large Data Systems, Proceedings of International Computer Symposium, 1975 (Vol.II), pp.157-165.

Lee,E.K.C. and Lee,E.Y.S. (1975), Development of a Data Dictionary/Directory Using a Data Base Management system, Second National Symposium on the Management of Data Element in Information Processing, Oct. 1975, pp.151-162.

Lefkovits,H.C. (1978), Data Dictionary Systems, Q.E.D. Information Sciences, Inc., Wellesley, Massachusetts, Nov. 1978.

Leong-Hong,B. and Marron,B. (1977), Technical Profile of Seven Data Element Dictionary/Directory Systems, NBS Special Publication 500-3, National Bureau of Standards, Department of Commerce, Feb. 1977.

Leong-Hong,B. and Marron,B. (1978), Database Administration: Concepts Tools, Experiences, and Problems, NBS SP 500-28, National Bureau of Standards, Department of Commerce, March 1978.

Lyon,J.K. (1976), The Database Administrator, John Wiley & Sons,Inc., 1976.

MSP Inc. (1976), DATAMANAGER: Fact Book, Management Systems and Programming Limited, Dec. 1976.

MBA Inc. (1977), Data dictionary/directory evaluation criteria, M.Bryce & Associates, 1977.

Plagman,B.K. (1977 a), Data Dictionary/Directory System: A tool for data administration and control, 22-01-02, Data Base Management, Auerbach Publishers Inc. 1977, pp.1-10.

Plagman,B.K. (1977 b), Criteria for the selection of data dictionary/directory system, 22-04-01, Data Base Management, Auerbach Publishers Inc., 1977, pp.1-10.

Software A G (1972), Data Dictionary Reference Manual, ADD-110-000, Software A G of North America, 1978.

SPARC/DBMS Study Group (1975), Interim Report, ANSI/X3/SPARC, American National Standards Institutes, 1975.

Towner,L.E. (1977), Integrated Data Base Development and Design Guide, Naval Intelligence Processing System Support Activity, AD A050468, Dec. 1977.

Tsubaki,M. (1979), Distributed Database and DD/D, DBMS-WG, Information Processing Society of Japan, Sept. 1979.

Tsubaki,M and Hotaka,R. (1979), Distributed Multi-Database Environment with a Supervisory Data Dictionary Databases, in Entity-Relationship Approach to Systems Analysis and Design, edited by P.P.Chen, North-Holland Publishing Company, 1980, pp.553-574.

Uhrowczik,P.P. (1973), Data Dictionary/Directories, IBM SYST. J., No.4, 1973, pp.332-350.

TABLE OF CONTENTS