Sci. J. Fac. Sci. Menoufia Univ. Vol. VIII (1994)

KNOWLEDGEABLE MULTIUSAGE DATA DICTIONARY

By
Dr. Hany M. M. Harb

Azhar University, Faculty Of Engineering,
Computers and Systems Engineering Department.

ABSTRACT

This paper discusses the design and implementation of a knowledgeable data dictionary. This knowledgeable dictionary is mainly designed to be associated with the relational database systems, but can be extended to be associated with other information systems like knowledge based systems. It can be installed as a general, stand alone package, or it can be integrated into an information system. It is based on relation as its data structure to be compatible with the relational database system, so both have the same SQL interface.

KEYWORDS
system analysis and design, data dictionary, knowledge based system, natural language.

1. INTRODUCTION

The data dictionary is essential in information systems and it may be called system catalog. Supporting an active on-line catalog is one of the Codd's twelve rules to determine what the relational database is [1]. The knowledgeable data dictionary (KDD) defines the data admissible to a system in a knowledgeable form. The KDD

names, classifies, represents, uses, and administers the system data and knowledge. It also defines the location of data within an information system (traditional system, database system, or knowledge based system), so it may be defined as an ideal path for access and retrieve in information systems. To support this function, this presented KDD contains also descriptions of report formats, screen menus, data records and files and input and output formats.

The KDD, as a system, has elements, environments, interactions between these elements, and the goals. So in general, KDD can be defined as a system organized of data, knowledge, documents, data and knowledge representation models, report formats, screen displays, input transactions, and procedures.

The KDD adds meaning to the stored data by storing its associated semantics. Knowledge has higher level of meaning because it represents information that can be potentially useful in future decision situations.

The KDD either as a stand alone or as an integrated information system is designed to satisfy the following goals:

- effective processing and management of data,
- flexible handle storage and retrieve of data,
- user friendly access, maintenance, and reporting
- efficient access and retrieval path of the associated knowledge based systems, if any.

To support the above goals, the KDD must have the following basic requirements:

- adequacy in documentation and in knowledge definition,
- data and knowledge representation efficiency,
- menu and/or forms facility, and

- structure modularity.

The major phases of KDD development process are discussed as follows. The determination of KDD goals and its basic requirements as previously stated is considered as the first phase. After goals and requirements determination, the KDD architectural design can be designed. Then, as the third phase, the KDD can be physically designed. In this phase, for example, dictionary entity format is determined in detail. The good design should consider maintenance, security, and enhancements.

As the fourth and last phase, the knowledge representation tool and the data structure are picked to build the KDD. The KDD operations to process and manage the dictionary are also defined in this phase.

The dictionary has to be carefully documented. Section 2 introduces the architectural design of the KDD. Section 3 discusses the KDD implementation and the frame as a knowledge representation tool to represent the KDD entities.

2. DESIGN

The KDD, as has been mentioned in previous sections, contains information concerning various objects that are of interest to the information system itself such as data, reports, schema, security, and so on as shown in Fig. 1. The KDD iterative process consists of analysis, design, and implementation phases as depicted in Fig. 2. In the analysis phase, the requirements and objectives are determined. The objectives and requirements of this presented dictionary are discussed in section 1.

The KDD keeps track of information about information system such as data structures (for example tables, columns, and indiees displays (forms), reports, users, views, etc. In this phase, only relational database system, as a type of information systems is considered. Other information system types are discussed as future research.

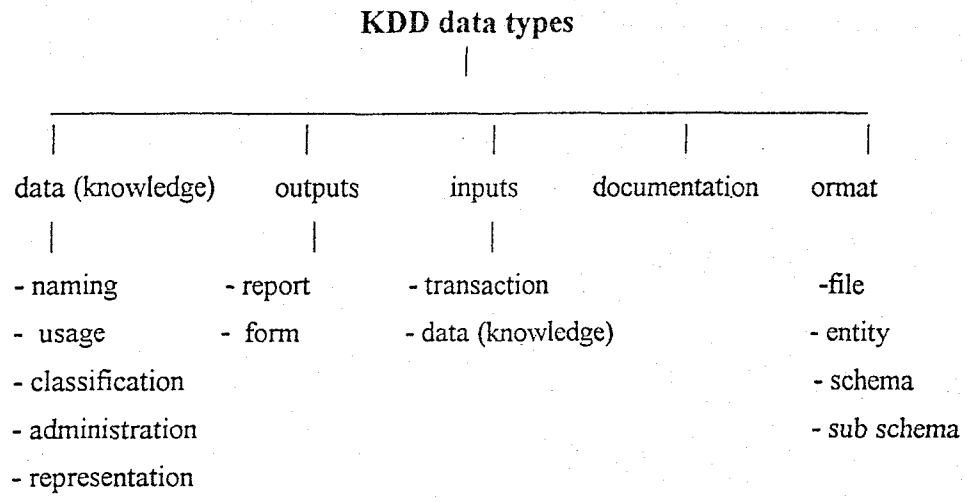

Figure (1): The KDD Knowledge types

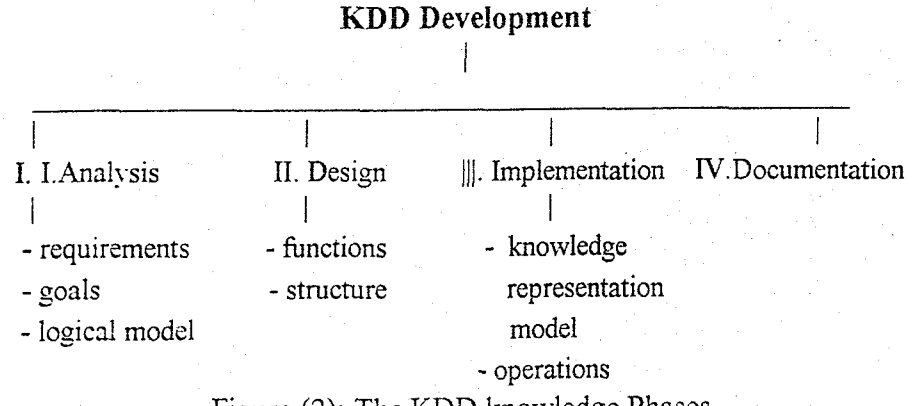

Figure (2): The KDD knowledge Phases

2.1 The KDD Data Structure

In the relational database system we should keep information about many things. These information are kept in tables to be compatible with the data structure of the relational database system itself. There are different tables supported by the KDD: KDD_table, KDD_column, KDD_table_column, KDD_index, KDD_index_column, KDD_column_column, KDD_user, KDD_form, KDD_report, KDD_menu, KDD_view.

Each table in the information has a row in KDD_table which keeps table name, table degree (number of attributes), cardinality (number of rows), table description, table security (who is authorized to do what is on this information system table mentioned in this row), table creator (table owner), last table structure alter time , last table data modification time (insertion, deletion, modification), and last table data retrieve time. Table security includes the authorization to add column(s), drop column(s), modify column(s) definitions (column name, column type, null acceptance), insert tuple(s), delete tuple(s), modify data values. Security field may contain any combinations of characters: a,p,c,i,d,m,r,x,y. If a character appears, then the corresponding right is allowed, otherwise it is prohibited. The meaning of each character is explained in Table 1.

Table 1: KDD_table table definition

attribute name	attribute meaning	attribute type
tname	table name	char
tdegree	table degree	numeric
tcard	table cardinality	numeric
tdesc	table description	char
towner	table owner	char
rtime	last retrieve time	date
atime	last alter time	date
mtime	last update time	date
tmode	security mode a: column add p: column drop c: column characteristics modification i: tuple insertion d: tuple deletion m: data modification r: data retrieve x: index add y: index drop	char

Each column in the database has one row in the KDD_column table with three attributes explained in Table 2. Different columns in different tables may have the same name. All the same name columns correspond to single row in this table, but different rows in the KDD_table_column.

Table 2: KDD_column table definition

field name	field meaning	field type
cname	database column name	char
cdesc	column description text	char
ctype	column type (integer, decimal,char....)	char
number_of_tables	number of tables in which the column name exists.	

A column may appear in more than one database table with different characteristics, so another table, KDD_table_column (Table 3), is needed to describe column appearance in each table to which it belongs. pk_or_fk states whether the column is a primary key (pk), a partial primary key (ppk), a foreign key (fk), partial foreign key (pfk), or a regular column (reg). The indexed field determines where there is an index wholly based on this column (wi), partially based on this column (pi), or it is not indexed (ni).

The column types differ from database management to another, but most of them are serial, integer, real, decimal, date, money, and interval.

Table 3: KDD_table_column table definition

field name	field meaning	field type
tname	table name	char
cname	column name	char
ctype	column type	char
pk_or_fk	key attributes (pk, ppk, fk, pfk,reg)	char
number_of_nulls	number of nulls	numeric
indexed	indexing attributes (wi, pi, ni)	char
rtime	last column data retrieve time	date
atime	last column type alter time	date
mtime	last column data modification time	date
cmode	column security mode c: column type modification m: column data modification r: column data retrieve	char

There are two tables to keep data about indiees. The KDD_index table (Table 4) expresses if an index accepts duplicate or unique values and the name of the table to which the index belongs. Since an index may be based on one or more than one column., the

columns on which the index is based are kept in another table, KDD_index_column table (Table 5).

Table 4: KDD_index table definition

field name	field meaning	field type
iname	index name	char
tname	table name	char
itype	index type u: unique index n: non unique index	char

Table 5: KDD_index_column

field name	field meaning	field type
iname	index name	char
cname	column name	char

A domain may be defined as a set of values of the same type from which the actual values appearing in attributes are drawn. One or more than one column may be drawn from a domain. The number of domains for a database system should be minimized. Generally, attributes drawn from the same domain can be compared. In other words, if two attributes drawn from different domains, then comparisons and hence joins and other operations involving these two attributes may not be legal . This is not always the case, since it still makes sense to compare different columns drawn from different domains. So, the KDD should support what may be called semantic override comparison (or domain check override) allowing that an operation is allowed even if involves a cross-domain comparison. Since domain concept is supported in most database systems, every

pair of attributes that cannot be compared are kept in the KDD_column_column table (Table 6).

Table 6: KDD_column_column table definition

field name	field meaning	field type
c1name	first column name	char
t1name	first table name	char
c2name	second column name	char
t2name	second table name	char

Every row defined in the information system corresponds to one row in the KDD_view table whose structure is expressed in Table 7. A view definition is expressed in vquery field. A view is a virtual table based on one or more base tables. The view does not exist in its own right, only its definition in terms of other tables (base table or other views) is stored in the catalog

Table 7: KDD_view table definition

field name	field meaning	field type
vname	view name	char
vquery	view definition in terms of other tables	char
vstatus	view status u: updatable view without check option c: up datable with check option n: not updatable	char
vowner	view creator	char
ctime	creation time	date
mtime	last modification time if updatable	date
rtime	last access time	date
vmode	view security mode	char

KDD_view table (query field). Changes to table(s) are automatically and instantaneously visible through the view based on these tables. If a view is updatable, then changes to this view are automatically and instantaneously applied to the tables on which the view is based. Not all views are updatable [2], in other words not all views are allowed to be inserted, deleted, or modified. If a view is updatable, modification and insertion operations can be checked to ensure that every updated or inserted tuple still satisfies the view defining condition. The vstatus expresses the view status as it is updatable, updatable with check, or not updatable at all.

The user identity is stored in KDD_user table as user name, user type, and any other comment about the user. The capability for each user is kept in KDD_user_entity. As has been shown, there are different levels of security: table mode (index mode, form mode, report mode) level, column mode level, and user mode level. The column mode level overrides the table mode level and the table mode level overrides the user mode level. For example, if a user is given access to a table, but the table is protected against this access, the user can access the table.

Table 8: KDD_user table definition

field name	field meaning	field type
uname	user name	char
utype	user classification dba: database administrator ord: ordinary user	char
cdesc	comment	char

Table 9: KDD_user_entity table definition

field name	field meaning	field type
uname	user name	char
ename	entity name	char
etype	entity type	char
	(table, column, index, form, report)	char
umode	user capability list on this entity	

Every form has a row in KDD_form table with form name, form creator, last modification date and time, and form protection mode. The form security modes are read, write, and execute as r,w,x respectively. For example if the fmode contains rx string, then the form can be read and executed.

Every report has a row in KDD_report table with report name, report creator, last modification date and time, and report protection mode. The report security modes are read, write, and execute as r,w,x respectively. The primary key of KDD tables are list in Table 12.

Table 10: KDD_form table definition

field name	field meaning	field type
fname	form name	char
fowner	form creator	char
mtime	last modification time	date
fmode	form security mode	char
	r: read	
	w: write	
	x: execute	

Table 11: KDD_report table definition

field name	field meaning	field type
rname	report name	char
rowner	report creator	char
mtime	last modification time	date
rmode	report security mode r: read w: write x: execute	char

Table 12: KDD tables primary keys

table name	primary key
KDD_table	tname
KDD_column	cname
KDD_table_column	tname,cname
KDD_index	iname
KDD_index_column	iname,cname
KDD_column_column	c1name,c2name
KDD_view	vname
KDD_user	uname
KDD_user_entity	uname,ename
KDD_form	fname
KDD_report	rname

3. Implementation

The KDD, as another option, can be installed and integrated into the information system. In this case, there is no need for the KDD administrator to submit directly any data. Instead, there is a KDD built-in interface between the information system and the KDD

which automatically updates the data in the dictionary as a consequence of each committed information system transaction. The menu driven user interface still exists, but only allowing the user to retrieve data from the dictionary. The KDD can be submitted as a stand alone package with SQL query language as an interface (so it may be different from the information system interface, if the information format ion system does not have SQL interface). This interface allows the information system administrator to manipulate the dictionary (or any other user that knows the dictionary password). To overcome this difficulty a user friendly menu driven interface is supplied as part of the KDD. For stand-alone KDD, the information system administrator himself submits all kinds of data to the KDD. The KDD has some rules to validate the integrity of the submitted data.

The menu-driven user interface has a layout as shown in Figure 3. The interface main menu has 5 options: Maintenance for data dictionary update, Info for information, SQL for SQL interface, Check for dictionary repair, and Exit for package quitting. The maintenance option allows the dictionary user to insert, delete, and modify any table values. A profile file may also be edited to customize the KDD environment. The user can highlight any menu option either by arrows or by a space bar. The highlighted option may be picked by enter, or any option may be picked by pressing its first letter. The KDD applies integrity rules as entity rule and referential rule. For example, a database table can not appear in any KDD table unless it exists first in KDD_table table, a column cannot appear in any KDD table unless it exists first in KDD_column table. This option is active only if the KDD acts as a stand-alone package since if the KDD is integrated into relational database system, the KDD data is updated automatically through another provided interface (hidden from the user).

Info option allows the user to get information about any table either database table or KDD table. The user may have the list of all tables and for any selected table (current table), the user may list its columns, indexes, and its security modes. For any selected column (current column), the user may check its type, its security modes, if it accepts null and if it is indexed. The user may consult and update the KDD tables through SQL option.

The Check option is a multi phase procedure checking the violations of integrity rules, and producing a list of primary and foreign keys. It also gives a warning message if a table name does not exist in any table but the KDD_table table, and if a column name does not exist in any table but the KDD_column table.

The user may customize the KDD environment by editing KDD_profile file. For example, the user favorite editor may be selected to edit SQL programs and KDD tables by assigning KDD_editor parameter to the full path name of the selected editor. KDD_home may be assigned to a directory to be a home directory in which all KDD tables will be resident. The full path name of a directory in which the KDD procedures and commands are resident may be assigned to KDD_path parameter. Other parameters are still available.

KDD_editor=full path of a user editor

KDD_home=full path of the KDD tables home directory

KDD_path=full path of KDD package directory

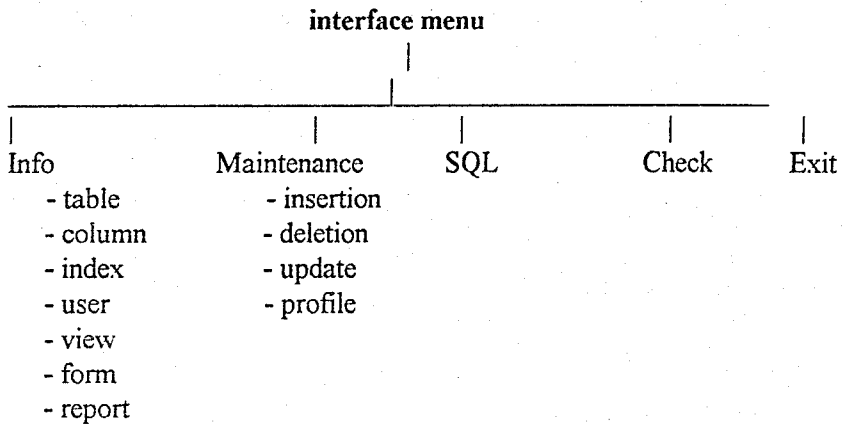

Figure (3): user interface menu

4. Extensions

The KDD may be extended, so it can be associated with knowledge based systems (KBS). The KBS includes a knowledge base, knowledge retrieval and storing capability, query capability, response reporting capability. The KBS can perform knowledge accessing procedures and in so doing it relies heavily on its associated knowledge dictionary. In other words, the KDD may assist the KBS parsing and understanding capability by translating the input message (query or knowledge) into another format to be more efficiently processed by KBS inference engine. Not only the syntax of the KBS internals, but also its semantics is saved in the KDD.

The extended KDD is made up of entries or items (an entry may correspond to one or more item), each entry consists of the entry identifier and its associated knowledge. There may be more than one entry type. Different types have different sets of associated knowledge; so the entry structure is type dependent. Examples of entry types are table, column, index, form, report and linguistic item types.

The KDD entry is expressed as an identifier plus any combination of three distinct domains: description, syntax, semantic domain. The description domain contains information to describe the entry such as name, usage, location, content type and security mode. Usage states the related entries, the reports use the item and so on. Location states where the item is located in the information system.

The security states the users who are allowed to access the item and the access type for each one. A example of an item description domain is shown in Figure 4. The syntax and semantic domain describe the syntax and the semantics of the item. These domains do not have meaning for all the entry types [2, 3, 4 ,5 ,8].

IDENTIFIER:	salary
NAMING :	wage
CONTENT:	numeric
LOCATION:	sa field
SECURITY:	user1, rw
	user2, rwm
COMMENT:	the employee's weekly salary
RELATED:	bonus, tax

Figure (4): The data item description domain

There may be more than one entry for the same item if it has more than one meaning . An item may be an element or a group. A data item is a group item if it consists of more than one elementary item. A group definition shows the data elements that make up the group and the relationship among them. A set of relational operators are used to define the composition of a group.

Frames as a knowledge representation structure or tables can be used to design the KDD, but the table is still preferred to keep SQL interface since most of the attached information systems are relational database systems. So, the tables discussed in the above section can be extended to satisfy the new requirements by including syntax and semantic information about the items.

CONCLUSION

This paper presents a complete design of a knowledgeable data dictionary. It was implemented and introduced as either stand-alone package or integrated into an information system. In this phase, it can only be integrated with a relational database system, but it can be extended to be compatible with other types of information systems. The KDD is based on table as the only data structure, so it has the same SQL interface with most of the relational database systems. A menu-driven user interface is built for the dictionary allowing the user to maintain, repair, and consult the dictionary data. The KDD environment may be customized through KDD profile feature.

REFERENCES

1. E.F. Codd, "Does Your DBMS Run by the Rules?," Computerworld, October 21, 1985.
2. A.L. Furtado and M.A. Casanova, "Updating Relational Views," Query Processing in Database Systems, New York, Springer Verlag 1985.
3. C.J.Date, "An Introduction To Database Systems," Volume I, Fifth Edition, Addison Wesley, 1990.

4. B. Gavish and H. Pirkul, "Computer and Database Location in Distributed Computer Systems," IEEE Trans. on Computers, Volume C-35, Number 7, July 1986.
5. H. R. Kanakia and F. A. Tobagi, "On Distributed Computations with Limited Resources," IEEE Trans. on Computers, Volume C-36, Number 5, May 1987.
6. S.C. Kak, "Data Security in Computer Networks," Computer, Volume 16, Number 2. Feb. 1983.
7. R.G. Canning, "A New View of Data Dictionaries," EDP Analyzer, Vol. 19, No. 7, July, 1981.
8. A. Barr, E. Afeigenbanm. "The Handbook of Artificial Intelligience," Vol. 1, William Kaufmann Inc., 1981.

قاموس بيانات معرفى متعدد الاستخدامات

يقدم البحث تصميما وتطبيقا معرفيا متعدد الاستخدامات و القاموس فى المرحله الحاليه متوائم مع تطبيقات قواعد البيانات العلائقية ولكن يمكن تطويعه بسهوله ليتوائم مع نظم معلوماتيه اخري مثل تطبيقات الذكاء الاصطناعى. ومن الممكن ان يعمل بصوره مستقله عن التطبيق المصاحب، ويعمل فى هده الحاله كموثق تفاعلى، او يكون متكاملا معه.