N89-16368

Rdesign: A Data Dictionary with Relational Database Design Capabilities in Ada*

Anthony A. Lekkos [1]
Teresa Ting-Yin Kwok

University of Houston, Clear Lake

1. Introduction

Data Dictionary is defined to be the set of all data attributes, which describe data objects in terms of their intrinsic attributes, such as name, type, size, format and definition. It is recognized as the database for the Information Resource Management -- to facilitate understanding and communication about the relationship between system applications and system data usage and to assist in achieving data independence by permitting system applications to access data without knowledge of the location or storage characteristics of the data in the system [Allen82].

The following are considered to be its primary objectives:-

1. To achieve control of the data resource, by providing an inventory of that resource. To enforce standards and validation.

2. To control the costs of developing and maintaining applications.

3. To provide for independence of metadata across computing environments, improving resiliency to the effects of hardware and software changes [Allen82].

Much of the importance of a data dictionary has been recognized, yet, little of it has been utilized to support an automated database design.

* Ada is a registered trademark of the U.S Government- Ada Joint Program Office.

[1] Supported by NASA/JSC-UHCL Ada-Beta site Contract.

A research and development effort to use ADA at UHCL has produced a data dictionary with database design capabilities. This project supports data specification and analysis and offers a choice of the relational, network, and hierarchical model for logical database design. It provides a highly-integrated set of analysis and design transformation tools which range from templates for data element definition or modification, spreadsheet for defining functional dependencies, normalization, to logical design generator.

2. The Data Dictionary with Database Design Capabilities

2.1 The Data Dictionary

The structure for the data dictionary is essentially relational in nature with the data element definition normalized to third normal form, while the related projects are kept in another relation. Further, the dictionary is furnished with the following facilities:-

Define --

creates a new data element entry in the data dictionary. A template is used to enter the data element name, type, size, range, description, validation rules, picture, intensity, display attribute, should the element type be enumeration, the enumeration list could also be entered.

Modify --

changes any data element specification created with "Define". If the data element name is changed, it creates a new data element under this new name -- essentially, it performs a "Copy" function under this circumstance. Again, a template is used for all field entries.

Search --

retrieves data element specifications from the data dictionary and displays them on screen. This differs from "Report" in that only data element names are displayed. A global search could be done by entering an "*". A list of names will be displayed on screen one page at a time.

Purge --

removes a data element from the data dictionary if the data element is used only in the current project. The element is not purged if it is used by other projects.

Transfer --

imports data element definitions from an external text file or exports data element definitions to an external text file. All of the elements could be imported or exported all at once, or they could be imported or exported individually, or they could be imported or exported according to projects.

Report --

lists in detail the data element definition for a single data element or a series of data elements. The listing could go to the terminal or to the system printer. If terminal is chosen as the output device, the data element definition will be displayed on screen one page at a time.

2.2 Functional Dependencies

Given a project with its own set of data elements, one can proceed to define functional dependencies amongst data elements. The following facilities are provided :-

Clear --

which clears out all previously defined functiona dependencies.

Spreadsheet --

data element names are displayed in rows and columns in a spreadsheet. Entering appropriate symbols in corresponding positions or "cells" will define functional dependencies amongst elements. Using the tab key or arrow keys, one can move around the cells. Should the arrow go beyond bounds the spreadsheet will move one column left/right or one row up/down dependent on the arrow key hit and its position. One can also move the spreadsheet one page at a time by pressing Function key 1, 2, 3 or 4 to go up, down, left or right.

The following symbols are used to define functional dependencies :-

==> means row element determines column element

<== means column element determines row element

KEY means element is a key, row element name should be the same name as column element

N/A means not applicable, row element name should be the same as column element name where the element is not a key

+=> means concatenation of row elements to identify column element

<=+ means concatenation of column elements to identify row element

To make the spreadsheet even more convenient to use, there are a few hidden keys :-

R = Refresh --
the screen is refreshed, in addition todisplaying symbols used to define elements functional dependencies, the complementary symbols are also displayed.

H = Help --
help can be invoked.

B = Beginning --
spreadsheet moves to the beginning of the list of data elements row-wise or column-wise.

E = End --
spreadsheet moves to the end of data element list row-wise or column-wise.

F = Find --
gets a particular data element which will be displayed in the middle of the list row-wise or column-wise.

T = Toggle --

Toggles the symbol, for example, pressing "T" at a place where it displays "N/A" will change the symbol to "KEY".

To update the functional dependencies, one only needs to blank out the entry, enter appropriate symbols or just toggle the symbols.

P = Print --

will print out the functional dependencies to the screen or to a file.

2.3 Database Design Generator

After the functional dependencies are defined, the normalization tool can be utilized to automatically normalize the relations in third normal form. Each table structure is displayed and a name should be given.

At this point, all the tables so created are in third normal form.

For application or implementation reasons, one may have to violate the rules for normalization or to keep certain relations not in third normal form. A "maintain-table" facility is provided so that a database designer can define his own table with its own set of keys and attributes. Moreover, he can rename a table, delete a table, delete certain keys or attributes in a table or add certain keys or attributes in a table. The system will not re-normalize these tables. There is one constraint, however, the keys and/or attributes had to be defined in data dictionary.

Again a spreadsheet is employed to define the relationships amongst relations, be it one-to-one, one-to-many, many-to-many or no-relations.

A refresh function will not be applicable in this case as the relationship between the row relation and the column relation may not be reciprocal.

A parent-child graph could be generated after the relationships are defined. The graph could be printed out to the system printer or to the screen.

The conceptual schema is then generated and output goes to the screen and a text file so that the designer can view it and make modifications if necessary. Since the data elements used are governed by the data dictionary, consistency, integrity and validity can be achieved easily.

Following is a SQL-type logical design interface so generated :-

```
Rem
Rem    SQL/DS Database Design Identification Section
Rem
Rem    Application : DIS
Rem
Rem    Date created: 6/6/86
Rem
Rem    ----------------------------------------------------
Rem    SQL/DS Database Tables Create Commands Section
Rem    ----------------------------------------------------

Create Table DEPT
     (DEPT_NAME            Char      (32) not null,
      BUDGET               Number     (7),
      DEPT_MGR             Char      (32),
      LOCATION             Char      (32));

Create Unique Index DEPT_INDEX on DEPT (DEPT_NAME);

Create Table EMPLOYEE
     (EMP_NAME             Char      (32) not null,
      DEPT_NAME            Char      (32),
      EMPLMNT_DATE         Char       (8),
      POSITION             Char      (32),
      SALARY               Number     (7));

Create Unique Index EMPL_INDEX on EMPLOYEE (EMP_NAME);

Create Table PROJECT
     (PROJ_NAME            Char      (32) not null,
      DEPT_NAME            Char      (32),
      CHARGE_NO            Number     (4),
      COMPL_DATE           Char       (8),
      PROJ_LEADER          Char      (32));

Create Unique Index PROJ_INDEX on PROJECT (PROJ_NAME);

Create Table EMP_PROJ
     (EMP_NAME             Char      (32) not null,
      PROJ_NAME            Char      (32) not null,
      CHARGED_HRS          Number     (4),
      EMPPROJ_DATE         Char       (8));

Create Unique Index EMPPROJ_INDEX on EMP_PROJ
     (EMP_NAME,PROJ_NAME);
```

3. User Interface

Much of the work for maintaining the data dictionary is done through a template, e.g., define and modify, user only needs to fill in the blanks, other work like defining functional dependencies and relationships amongst tables is done through a spreadsheet. The whole system is menu driven with the first two rows of the screen dedicated to commands. To go up to the command line, one only needs to press <F2>. One can then use Tab, or arrows to move across the command line. A <return> selects th command. The 24th row on the screen is dedicated to function key explanation while the 23rd row is used for message line and the 22nd row is used for prompt line. The rest of the screen will be used for display or for the template. Help could be invoked throughout the screens.

References

[Allen82] Frank W. Allen, Mary E. S. Loomis, Michael V. Mannino "The Integrated Dictionary/Directory System" , ACM Comp. Survey,14:2, 1982.

[Goldstein85] Robert C. Goldstein "Database :Technology and Management John Wiley and Sons, Inc., 1985.

[Curtice84] R.M. Curtice "IRMA: An Automated Logical Data Base Design and Structured Analysis Tool", IEEE Database Eng. 7:4, December 1984.

[Reiner86] David Reiner, Gretchen Brown, Mark Friedell, et al, "A Database Designer's Workbench" submitted to Dijon ER Conference, 1986.

[Bjornerstedt84] A.Bjornerstedt and C. Hulten "RED1: A Database Design Tool for the Relational Model of Data", IEEE Database Eng. 7:4, December 1984.