North-Holland
Microprocessing and Microprogramming 25 (1989) 375 – 380

Automated Normalization Tool

T. Welzer, I. Rozman and J. Györkös

UNIVERSITY OF MARIBOR,
FACULTY OF TECHNICAL SCIENCES, Smetanova 17, P.O.BOX 224,
YU-62000 MARIBOR, Yugoslavia

In our contribution the Automated Normalization Tool named **ANT** and some experience with its development are described. The base of this tool is the normalization theory which is built on the concept of normal forms. ANT has an active guidance through the four main possibilities and is intended for good design of relational database. In this paper the Theoretical Backgrounds and Fundamental Aspects of ANT with a Short Description of Possibilities are described. For the implementation of ANT a nonprocedural language PROLOG is used. The ANT is also possible as an optional part of a Computer Automated Software Engineering Tool named **ASET** (Automated Software Engineering Tool), which is intended for the development of the program systems.

1. INTRODUCTION

One of the most important modules for various computer applications is **database**. By G. Wiederhold [5] the database presents a feasible, simplified and validated model of the real world. According to the source Date [3] the most important advantages of database are:

- **Compactness**: No need for possibly voluminous paper files,
- **Speed**: The machine can retrive and change data far faster than a man
- **Currency**: Accurate, up - to - date information is available on demand at any time.

Three logical database models (hirearchial, network and relational) are most known and usually used by users. The **relational** database is more and more often used instead of the first two databases. Today it is widely accepted, not only in academic circles but also in the market place and it is successfuly gaining ground on all fields of application as well. The advantage of relational database is above all in its simple and user friendly **transfer** from the paper to the computer. (Date [3]: The user of relational database sees data as tables and nothing but tables).

Successful work with the relational database can be ensured only by a good design of the base. This design should be ensured by a multitude of rules, called **normal forms**. Normal forms (numeros of them have been defined) are rules on the joins of attributes into relations in which logical dependences are taken into account. After source Date [3], Codd[1] defined the first, the second and the third normal form (NF) originally. Subsequently the Boyse/Codd normal form (BCNF), the fourth NF and the fifth NF (also known as projection/join normal form) were defined. Because of complexness of all these rools the design of relational database should be aided by corresponding tool (Welzer [6]) which should guide the user from decision to decision.

1.1 Theorethical Backgrounds of ANT

As a theoretical background of ANT (some basic definitions from relational database theory) the source Alagić [1] was chosen.

1. A **relational database** is a time-varying set of relations. R(A1,A2,...,An) is a relation over the sets D1,D2,..,Dn if and only if the subset of Descartes product is as follows:

$$R \subseteq \{D1 \times D2 \times \ldots \times Dn\}$$

D1,D2,...,Dn mean the **domains**[2] of attributes. The attributes A1,A2,...,An describe the structure of the relation which is called **relational schema**.

2. **Relational algebra** is a simple formal language which enable data manipulation. It consists of two groups (originally defined by Codd [8]) of operations over relations:

- the traditional set operations union, intersection, difference and cartesian product;
- the special relational operations selections, projections, join and division.

For our further work, operation of projection is of main importance.

Let R(A1,A2,...,An) be a relation and X and Y subsets of the set of atributes . The relation R(A1,A2,...,An) can be written as follows: R(X,Y). The projection of the relation R(A1,A2,...,An) according to attributes X is denoted as : R[X] and defined as :

$$R[X] = \{x \mid \exists y: (x,y) \in R(X,Y)\}$$

3. The **functional dependence** X --> Y exists in the relation R(A1,A2,...,An) if and only if it is valid for R[XY] in every moment of time that in fact R[XY] is a function R[X] --> R[Y]. R[XY] denotes the projection of the relation R(A1,A2,...,An) according to attributes from subsets X and Y.

The functional dependence X --> Y is said to be **full** if and only if for every proper subset X' (X' $\subseteq$ X) it is **valid that X' -/-> Y. If for some proper subset X' functional dependence X' --> Y exists then the functional dependence X --> Y is called a partial.**

4. A subset X of a set of attributes of the relation R(A1,A2,...,An) is said to be the **key** of the relation R(A1,A2,...,An) if and only if the following two conditions are fulfilled:

- X functionally determines all attributes of the relation R(A1,A2,...An), X --> Ai for i = 1,...,n;
- no proper subset X' of set X posseses this characteristic; for X' X' -/-> Aj is valid 1<j<n.

Such key(s) is/are called **candidate key(s)**. Among all key(s) (candidate keys) of the relation R(A1,A2,...,An) one is usually chosen to serve as an identifier of entities[3] represented by the tuples[4] of R(A1,A2,...,An). This chosen key is called the **primary key**.

5. The relation R(A1,A2,...,An) is in the **first normal form** if and only if the values in the domains are atomic for every attribute A in the relation R(A1,A2,...,An).

6. Let X be a set of all attributes of R(A1,A2,...,An), which are not a part of the key of the relation R(A1,A2,...,An). It is said that the relation R(A1,A2,...,An) is in the **second normal form** if and only if each attribute from X is **fully functionally dependent** on each key of the relation R(A1,A2,...,An).

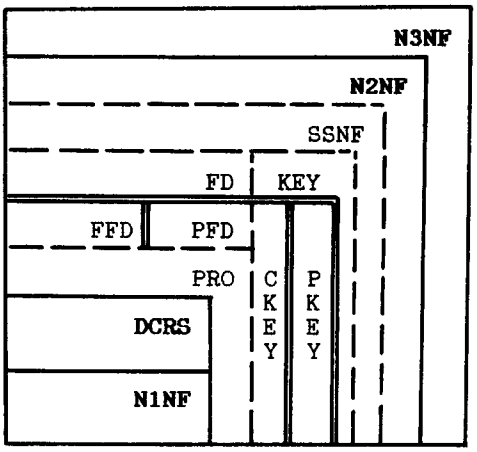
2. FUNDAMENTALS of ANT CONCEPT

ANT **A**utomated **N**ormalization **T**ool is a menu driven tool with four main possibilities (see Figure 1):

- Defining the Contents of Relational Schema,
- Normalization into the First Normal Form,
- Normalization into the Second Normal Form,
- Normalization into the Third Normal Form.

In the frame of the single possibilities (see N2NF on Figure 1) an active guidance through different modules (PRO, FD, KEY, SSNF) and submodules (FFD, PFD, PKEY, CKEY) is possible.

Figure 1 shows the exact scheme of ANT. In the mean of the statement, that the **hirearchy** of normal forms enables that the relation which satisfies the requirements of some normal forms also satisfies the requirements of all **lower - order** normal forms, the four main possibilities of ANT are graphically presented on Figure 1. From this Figure is also seen that the N2NF possibility is as a whole decomposed into several modules and submodules. The basic module of N2NF is the **projection** modul named **PRO** and the rest modules (FD,KEY, SSNF) are built around this. Similar decomposition is also found out for the N3NF.



The Explanation of the Acronyms:

DCRS - Defining the Contents of Relational Schema
N1NF - Normalization into the first Normal Form
N2NF - Normalization into the second Normal Form
N3NF - Normalization into the third Normal Form
PRO - Projection
FD - Functional Dependency
FFD - Full FD
PFD - Partial FD
CKEY - Candidate key
PKEY - Primary key
SSNF - Stating of the second normal form

Figure 1. Graphical presentation of ANT

2.1 Short Description of ANT Possibilities and Modules

2.1.1 Defining the Contents of Relational Schema

DCRS enables defining of the relational schema and its contents and if necessary also the normalization into the first normal form. Data entry is very simple and the users enter data as tables[5] (see citation Date from Chapter 1). Data should be entered into an existing file (this does not mean update of attributes and data) or in to a new one. The normalized entry table (in the first normal form) is shown on a screen by termination of DCRS possibility.

2.1.2 Normalization into the First Normal Form

This possibility enables that according to the definition 5 from Chapter 1.1 an unnormalized table is translated into the first normal form. For the normalization of such relation the following steps are done:

- checking the exsistence of redundant lines and excluding them,
- finding out whether the values of individual attributes in the domain are recorded as sets or lists. If such records exist, they must be translated into a corresponding form.

The result of the described possibility is shown on a screen as a normalized table which is also written on a file.

2.1.3 Normalization into the Second Normal Form

Using this possibility we should avoid the irregularities (anomalisms) of data input, delete and update. At first the relation must be checked in the light of the second NF. The relation which is found out that it is not in the second normal form, should be decomposed into several tables that would meet the condition for the second

normal form. Both steps, test and decomposition should be realized with help of modules and submodules numbered in previous chapter. The most important of them is the projection module named **PRO**. According to the definitions 2,3,4 and 6 from the Chapter 1.1, PRO is the base for stating the full and partial functional dependence as well as for determination of candidate keys and primary key and not at last also for stating the existence of the second normal form.

PRO. This module enables that according to the definition (def. 2 Chapter 1.1) a new table is created. The result of the named module is the table into which the required columns were translated and the redundant lines were deleted.

FD. According to the Chapter 1.1 (definitions 3,4 and 6) and Figure 1 the functional dependence is build on module PRO. On the other side the FD module is the base of modules KEY and SSNF.

The existence of functional dependence between sets X and Y is stated by checking the contents of a part of relational schema that form the mentioned sets. For these intentions the module PRO is used first to design the relation R[XY]. Then in this relation the condition R[X] --> R[Y] (definition 3 from Chapter 1.1) is fulfilled. According to the mentioned definition the functional dependence can be **full** or a **partial** one. To state this caracteristic the submodules **FFD** and **PFD** are used.

KEY. In a case that the key is not determined, this module and its submodules are used in a frame of possibility N2NF or independent. The submodule **CKEY** enables that according to the definition of candidate key (Chapter 1.1) all candidate keys in the table are chosen. Between them we can determine a primary key. In the case that the primary key is known the KEY module is over leapt.

SSNF. Before the normalization into the second normal form takes place it must be stated if the normalization is needed. The test is executed with help of modules and submodules described previously.

3. APPLICATION OF PROLOG IN THE RELATIONAL DATABASE THEORY

PROLOG **Pro**graming in **Log**ic, a nonprocedural language is used for the implementation of ANT. An ascendancy of this language over the languages of the fourth generation is according to the source Marcus [4] in the **built-in** database of PROLOG and also in the fact that the program is not defined as a sequence of steps. The PROLOG program is defined as a description of relations between objects.

PROLOG's built-in database ensures a basic mechanism for data storage and data access. The notation of relation is a simple one. The data obtained from the relation (table) are recorded in the form of PROLOG's facts (Clocksin [2]), consisting of predicates and attributes (these are not attributes from relational database theory). The name of relation is written in predicate ; the values of attributes (the contents of relational schema) obtained from the table are written in attributes. After the input of all data into the relational schema (this is not data entry from DCRS) we can see that a record on the screen is equivalent to the record on the paper (see citation Date from Introduction).

3.1 Implementing of ANT in PROLOG

The facts from the previous Chapter about notation of relational database in PROLOG are the base for implementation of our tool, which is programed in several PROLOG programs. Some of them have general purpose like **read_in** (read_in enables reading from the keyboard, that means users friendly communication with the tool), but most of them are specially written for ANT. The basic idea of these programs is in using of PROLOG's structure named list. According to the source Clocksin [2] the list is a very common data

structure in non-numeric programming. It is an ordered sequence of elements (constants, variables, structures - which of course includes other lists). In the structure of lists the relations are translated and then with the different operations over the lists the requested goals are achieved.

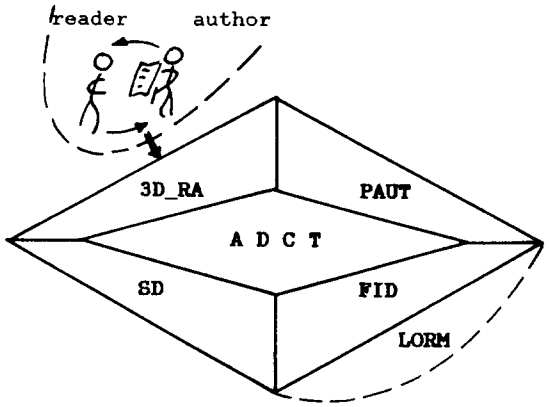
N1NF. The program which enables this possibility is based on checking the elements in the structure of lists. When the number of elements in individual lists is found out the paralel laying elements are joined into lines which are then transmitted to the screen in their normalized form.

DCRS. This program is in nature the N1NF program supplemented with a part which with help of PROLOG's built-in predicates enables writting of the entry data in the file.

N2NF. The program is composed from several programs. Some of them are parts of already described programs. According to the facts from Chapters 1, 1.1, 2 and 2.1 the PRO program is the most important part of N2NF. The sequential numbers of selected attributes are chosen at first in order to find out all possible solutions. This program is then a base for FD, KEY and SSNF programs.

4. ANT AS A PART OF ASET

ANT is also possible as a part of a CASE tool named ASET (Automated Software Engineering Tool). The ASET is described in detail by Györkös [9]. It is foreseen for the developing of the program system, realized with the procedural languages and optionally with database support. The most important feature of the ASET is the covering of most phases of software cycle (for definition of software cycle see ANSI/IEEE Std. 100-1984 [7]). Special attention is paid to the dialogue with the user. Figure 2 shows which basic building blocks make the ASET and on Figure 3 the decomposition of 3D_RA block in sub-blocks is presented. Figure 3 also shows that ANT is an optinal part of sub-block named Information Analysis.



The Explanation of the Acronyms:
- **ADCT** - ASET Dictionary
- **SD** - Structured Design
- **PAUT** - Prototyping and Unit Module Testing
- **FID** - Final Document
- **LORM** - Library of Reusable Modules
- **3D_RA** - Three Dimensional Requirement analysis

Figure 2. Basic Building Blocks of ASET

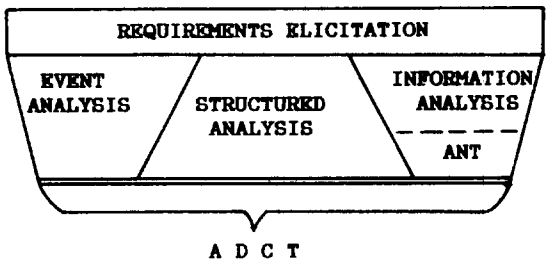


Figure 3. Decomposition of 3D_RA Block

For ANT - the ASET Dictionary (Figure 2), called ADCT is the most important building block of ASET. For this fact the following reasons are given:

- the dictionary contains a composition of entities,
- ADCT entity form is used to model relational schema,
- the normalized entities are used again in ADCT.

After an active inclusion of ANT into ASET, the four possibilities described

in the previous chapter may be performed. The normalized entities are transfered in ADCT by means of return from ANT into ASET.

5. CONCLUSION

The described ANT concept supports the work of database administrator. The design of different relational schema , between which the final normalized solution is chosen (for work in the real evironment) is enabled. For smaller databases the PROLOG environment could be used. In this case the simple query language schould be built. All in this paper mentioned programs are performed by PROLOG interpreter on the main frame computer VAX 8800. The language PROLOG was chosen because of its built-in database and simple definition of rules (records of program). The deficiency of this choice is in the slowlyness, in too little memory capacity and in less efficient arithmetics. Some of this deficiency should be avoided with the use of PROLOG compiler. We will also continue with our work on this project. The goals of further work are:

- optimization of programs,
- realization of programs for normalization into the third normal form and
- establishment of simple query language.

FOOTNOTES

1 In the CACM 13, No 6 (June 1970) his paper "A Relational Model of Data for Large Shared Data Banks" was published. This paper represents the begining of relational database theory.

2 A **domain** is a pool of values (Date [3]) from which one or more attributes draw their actual values.

3 After source Date [3] the term **entity** is widely used in database circles to mean any distinguishable object that is to be represent in the database.

4 A **tuple** corresponds to a row of the table.

5 Table is the synonim for relation.

REFERENCES

[1] Alagić, S., Relational Database Technology (Springer - Verlag, New York, 1986).

[2] Clocksin, W.F. and Mellish, C.S., Programming in Prolog (Springer - Verlag, 1984).

[3] Date, C.J., Database Systems (Addison - Wesley, 1986).

[4] Marcus, C., Prolog Programming (Addison - Wesley,1986).

[5] Kaiser, G.E., Barghouti, N.S., Feiler, P.H. and Schwanke, R.W, Database Support for Knowledge-Based Engneering Environments, IEEE EXPERT, Vol.3, No.2, (1988), pp. 18-32.

[6] Welzer, T., Rozman, I. and Gyorkos, J., The Normalization of Relational Database, INFORMATICA Journal of Computing and Informatics 3, (1988), pp. 12-15.

[7] ANSI/IEEE Std. 100-1984, in: Jay, F.(ed), IEEE Standard Dictionary of Electrical and Electronics Terms (IEEE, New York, 1984).

[8] Codd, E.F., Relational Completness of Data Base Sublanguages, in: Rustin, R.(ed) Data Base Systems, Courent Computer Science Symposia Series, Vol.6 (Prentice - Hall, 1972).

[9] Györkös, J., Rozman, I. and Welzer, T., The Concept of an Efficient Computer Aided Software Engineering Tool, in: Procedings IFAC88 (Pergamom,1988) will be published.