Full paper in Proceedings of the 16th Annual NACCQ, Palmerston North, New Zealand, July, 2003 (eds) Mann, S. and Williamson, A. www.naccq.ac.nz

Logical Relational Datamodelling through Normalisation by Synthesis

J.P.Nijsse
ComTech Department
Wanganui UCOL
p.nijsse@ucol.ac.nz
R.J.Whiddett
Massey University
C.F.Atkins
Nelson Marlborough Institute of Technology

ABSTRACT

This paper describes a step by step method to develop a logical relational datamodel. The method used is an adapted version of Normalisation by Synthesis, which is easy to apply, methodical and non-subjective.

1. INTRODUCTION

This paper will explain to the reader a version of Normalisation by Synthesis, using non-mathematical terms as far as possible. The aim of this paper is to provide teachers of relational database design with an alternative approach, which is easy to apply, methodical and non-subjective.

Normalisation by Synthesis is a method to create a logical relational database schema from a given set of attributes. This set of attributes is known as the Universal Relation (U). It is assumed that for the database under consideration each relation comprising the logical schema is a projection of the Universal Relation.

The merits and demerits of assuming a Universal Relation have been discussed in various papers (Kent 1981, Ullman 1983). One of the objections raised in Ullman (1983) is the fact that attribute names are unique in the Universal Relation. Hence the same attribute name occurring in different relations of the logical schema must mean the same thing (and have the same domain), since they are projections of the same attribute of U. Conversely, if the analysis process identifies attributes, which have different names, but have the same domain and are used in the same context, they must be given the same name in the Universal Relation. This objection, together with others, are countered in Kent (1981).

The application of the method requires a thorough understanding of the attributes of U, and how these attributes are used in the database. Provided this preliminary work is done, the results of this design method using the Universal Relation are very satisfactory.

Normalisation by Synthesis (Bernstein 1976) uses the Universal Relation as a carrier of functional dependencies. Each attribute in U is analysed with respect to its functional dependencies, and attributes with the same functional dependencies are grouped together to form the relations. This is in itself a simple process and purely syntactic in nature, and takes no account of semantics, (the semantic study of attributes and their naming being part of the preliminary process). Applying this process will put the resulting relations into at least Third Normal Form (3NF). In fact, it puts the relations into Elementary Key Normal Form (EKNF), which lies between 3NF and Boyce-Codd Normal Form (BCNF)(Zaniolo 1982, Date 1995).

The method described in this paper is an adaptation of this process. By adding a few more restrictions and extending it to include dependencies other than functional ones, the method produces a better design. The next two sections of this paper describe the process of transforming the Universal Relation into a set of normalised relations. The following sections then demonstrate the process by applying it to two examples. The paper concludes with a discussion of the uses of this technique.

2. THE METHOD

The basic steps used in this method are set from below. Practical examples of each step is given in section 4 of this paper.

1. For each attribute of the Universal Relation find suitable determinants. A determinant can consist of more than one attribute. See the next section for an explanation of 'suitable' determinant.

2. Group together all attributes sharing the same determinant to form the relations, the determinant becoming the primary key. A logical relational schema results, showing only those relations having attributes other than the primary key.

3. Group together all primary keys in the above schema, plus any attribute not represented in the schema. Eliminate duplicates. This will make a list of key attributes. Reduce this list by removing any attribute that is functionally dependant on other key attributes.

4. Examine any single key attribute that is not a primary key of a relation in the schema. A single attribute relation may be included in the schema if the relation is not trivial.

5. Examine any combination of two or more key attributes that is not a primary key of a relation in the schema. If there is a dependency between the key attributes, and the relation provides meaningful information, this key-only relation should be included. If the key consists of three or more attributes, normalize the relation to 5NF.

3. SUITABILITY OF DETERMINANT

This section discusses the protocol that can be used to identify suitable determinants.

Within this paper the following notation is used:

A à B or B ß A

means attribute A determines attribute B or attribute B is functionally dependent on attribute A. A, B E U attribute A is called the determinant of attribute B

A ßàB

A and B determine each other

A /à **B**

A does not determine B

A suitable determinant should conform to the following requirements:

a As the determinant of an attribute becomes the primary key of a relation, it should fulfil the basic requirements of a primary key, ie Applicability, Uniqueness, Minimality and Stability [6]

Applicability: A value for the determinant must exist for every instance of the attribute it determines within the relation.

Uniqueness: Inherent in the definition of determinant.

Minimality: A determinant must not include attributes beyond those required to ensure uniqueness.

Stability: For a particular instance of the attribute the determinant should keep the same value for the duration of the database.

b Each attribute comprising the determinant must be a primary key of a related relation. This relation may be trivial and need not appear in the schema. This will ensure that anomalies such as overlapping and split foreign keys [6] will not occur.

c A determinant must not transitively determine an attribute. This occurs when:

i Aà B à C and B */*à A. In this case B is the determinant of C and not A. The exception is when instances exist where B is null and A and C have values other than null. In that case both B and A are determinants of C. This is necessary to avoid chasm traps [6], [7].

ii A,DßàA,BàC, DàB and B */*àD. A,D and A,B are overlapping candidate keys. In this case A,B should be used as the determinant of C, and not A,D.

d. If an attribute has two or more determinants, and the determinants determine each other, then only one must be selected. The ones not selected must

not be used as a determinant, or part of a determinant, elsewhere.

The exception to this rule is in cases where you choose to make each determinant the primary key of a separate relation. 3d then no longer applies, but 3f applies.

e. If an attribute has two (or more) determinants such that one determinant is not functionally dependent on the other, then all must be selected.

f. In one-one relationships, the primary key of the optional part of the relationship determines the primary key of the mandatory part. Where supertypes/ subtypes are involved, this means that the primary key of the subtype determines the primary key of the supertype.

4. EXAMPLE (COURSE MARKIING SYSTEM)

The following example illustrates the method outlined in Section 2, and the choice of suitable determinants outlined in Section 3.

(Note: Step1, Step 2 …. Relates to the steps outlined in section 2. 3, 3e, …. Relates to the requirements set out in section 3)

The following attributes make up the Universal Relation:

Student ID, Student name, unit, unit description, credits, assessment no., assessment type, mark, grade, student address, lecturer ID, lecturer name, dept ID, dept Name, dept head

Explanations of some of the attributes are:

Credits are the number of credits awarded for the successful completion of a unit

Assessment no is an integer (1,2,3…) which is used to distinguish one assessment from another for a given unit

Mark is the mark awarded to a student for a particular unit assessment

A grade (A,B,C…) is awarded to a student on completion of a unit

Assessment type denotes what kind a particular unit assessment is (ie test, assignment, case study…)

The following constraints apply:

- A student may take more than one unit
- A lecturer works for only one department
- The delivery of a unit may be shared by more than one lecturer, but a particular unit assessment is not dependent on the lecturer

Step 1: For each attribute above, find suitable determinants.– For suitability see Sect.3

Note that some attributes may not have any suitable determinants. If the attribute is itself a suitable determinant, then the determinant should be left blank.

If this is not the case, then a suitable determinant needs to be added to the Universal Relation.

Note also that an attribute may have more than one determinant – Rule 3e.

(see over page)

Step 2: Group together all attributes sharing the same determinant.

The following relations are obtained:

(lecturer ID, lecturer name, deptID)

(student ID, student name, student address)

(unit, unit description, credits)

(dept ID, dept name, dept head)

(unit, assessment no, assessment type)

(student ID, unit, assessment no, mark)

(student ID, unit, grade)

Step 3: Make a list of all key attributes of the relations in Step 2.

The key attributes are:

lecturer ID, student ID, unit, dept ID, assessment no

Remove those that are functionally dependant on other key attributes. Dept ID can be removed from this list as lecturer ID à dept ID.

The reduced list is lecturer ID, student ID, unit, assessment no.

Step 4:

'assessment no' is the only key attribute which is not a primary key of the schema. The relation is valid, but trivial so should not be included.

Step 5:

The only combination which could contribute useful information to the schema is 'lecturer ID' and 'unit', as this tells us which lecturers teach which units, information which cannot be obtained from the schema. If there is an identified need for this information, then the relation (lecturer ID, unit) should be added to the schema.

student ID ←		Left blank as student ID has no determinant and is itself a suitable determinant.
student name ←	student ID	Satisfies requirements in 3
unit ←		Left blank
unit description ←	unit	Satisfies requirements in 3
credits ←	unit	" " " "
assessment no ←		Left blank
assessment type ←	unit, assessment no	Satisfies requirements in 3
mark ←	student ID, unit, assessment no	" "
grade ←	student ID, unit	" " " "
student address ←	student ID	" " " "
lecturer ID ←		Left blank
lecturer name ←	lecturer ID	Satisfies requirements in 3
dept ID ←	lecturer ID	" " " "
dept name ←	dept ID	Although lecturer ID also determines dept name, it is transitive – Rule 3c
dept head ←	dept ID	Rule 3c as above

Step 1

Note that the same result could have been obtained in a more formal manner if a relation of the reduced list in Step 3 is formed.

(lecturer ID, student ID, unit, assessment no)

By applying Fourth Normal Form to this relation, the net result is the addition of the relation (lecturer ID, unit) to the schema.

5. NORMALISATION EXCEPTION

In most cases correct application of the method leads to logical schemas that are fully normalised, ie free from insertion, deletion and modification anomalies. There are situations where this is not the case. However, in these cases the resulting relations cannot be fully normalized, and the method still gives the better schema, as the following example illustrates.

Consider the following attributes:

customer no, customer name, salesperson ID, salesperson name, branch ID, branch name, activity details

Note: 'Activity details' denotes that amount of business a customer has done with a branch.

The following constraints apply:

- A salesperson works for only one branch
- A customer is allocated only one salesperson per branch

Step 1:

Determinants are:

customer no ←	
customer name ←	customer no
salesperson ID ←	customer no, branch ID
salesperson name ←	salesperson ID
branch ID ←	salesperson ID
branch name ←	branch ID
activity details ←	customer no, branch ID

Step 2:

The following relations are obtained:

(customer no, customer name)

(salesperson ID, salesperson name, branch ID)

(branch ID, branch name)

(customer no, branch ID, salesperson ID, activity details)

Steps 3, 4 and 5 yield nothing further.

The above logical schema is not in BCNF, as in the last relation Salesperson ID à branch ID and salesperson ID is not a candidate key of the relation.

To put the relation into BCNF, the last relation needs to be changed to:

(customer no, salesperson ID, activity details)

By making this change, the schema is now in BCNF, but it is not in DKNF, since the constraint that a customer can only be allocated one salesperson per branch is no longer enforced, Moreover, the key customer no, salesperson ID is unstable, and transitively determines activity details. (3 c ii)

(customer no, salesperson ID ß à customer no, branch ID à activity details, salesperson ID à branch ID and branch ID *I*à salesperson ID)

Since it is clear that neither schema is fully normalised, and therefore could lead to insertion, deletion and update anomalies, a choice needs to be made as to which schema can more easily cater for these anomalies. The considerations above, together with a study of the anomalies, lead to the conclusion that the schema given in Step 2 is to be preferred (non BCNF model).

6. CONCLUSION

As illustrated by the examples, the adaptation of the process of Normalisation by Synthesis, which is presented in this paper, yields consistently good results, provided that a clear understanding of the attributes in the Universal Relation is acquired. This means that the System Definition and Requirements Collection and Analysis stages of the Database Development Cycle have to have been completed prior to this process. Like Normalisation by Decomposition (which is the normalisation method usually described in textbooks on databases Date 1995, Simsion 1994), Ricardo 1990)), this adaptation of Normalisation by Synthesis can be used in conjunction with more semantic modelling techniques to check on the correctness of the logical relational model. (Semantic modelling techniques like ER, NIAM, ORM are described in standard textbooks). Because the method is for the greater part mechanical and methodical, it can easily be automated up to EKNF. Also, because it is not subjective and can be described in non-mathematical terms, it is more easily understood by people with limited mathematical knowledge.

This method has been demonstrated to third year degree students, who were already familiar with Entity Relationship Modelling and Normalisation by Decomposition. Anecdotal evidence from the students indicates that they found this method easy to understand and apply, with less chance of error. They also found it far less subjective than the ER method, with hard and fast rules to follow, which students appreciate. Using the method in conjunction with the top-down modelling process would help greatly in promoting a thorough understanding of the attributes. Further examples illustrating the use of the various requirements of Section 3,- as well as a study of the anomalies in Section 5, are available elsewhere (Nijsse 2003).

Whether this process is used for educational purposes, or by database designers, it provides a valuable tool in the design of relational databases.

REFERENCES AND BIBLIOGRAPHY

BERNSTEIN P.A. (1976) Synthesizing Third Normal Relations from Functional Dependencies. In *ACM Transactions on Database Systems* 1,4 (Dec 1976) pp 277-298

CONNOLLY, T., BEGG, C., STRACHAN, A. (1999) *Database Systems – A Practical Approach to Design, Implementation, and Management* 2nd Edition Addison-Wesley Longman Limited

DATE C. J. (1995) *An Introduction to Database Systems* 6th Edition Addison-Wesley Publishing Co Ltd

HALPIN T. A. (1995) *Conceptual Schema and Relational Database Design* 2nd edition Prentice Hall, Australia

KENT W. (1981) Consequences of Assuming a Universal Relation. In *ACM Transactions on Database Systems* 6, 4 (Dec 1981) pp 539-556

McFADDEN F. R., HOFFER J.A., PRESCOTT M.B. (1999) *Modern Database Management* 5th edition Addison Wesley Longman

NIJSSEN G. M., HALPIN T.A. (1989) *Conceptual Schema and Relational Database Design: A Fact Orientated Approach* Prentice Hall of Australia Pty Ltd 1989

NIJSSE J.P. (2003) *Relational Database Design: A Practical Method using Normalisation by Synthesis*. Research Report, Department of Information Systems, Massey University, Palmerston North.

RICARDO C. (1990) *Database Systems: Principles, Design and Implementation* MacMillan Publishing Co

SIMSION G. (1994) *Data Modeling Essentials – Analysis, Design and Innovation* International Thomson Computer Press 1994

ULLMAN J.D. (1983) On Kent's 'Consequences of Assuming a Universal Relation' In *ACM Transactions on Database Systems* 8, 4 (Dec 1983) pp 637-643

ULLMAN J.D., WIDOM J. (1997) *A First Course in Database Systems* Prentice Hall Inc.

ZANIOLO C. A New Normal Form for the Design of Relational Database Schemata. In *ACM Transactions on Database Systems* 7, 3 (Sept 1982) pp 489-499