# QUAESTIONES INFORMATICAE

Vol. 1 No. 2 September, 1979

# Quaestiones Informaticae

**An official publication of the Computer Society of South Africa**
**'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Afrika**

## Subscriptions

Annual subscriptions are as follows:

| | SA | US | UK |
|---|---|---|---|
| Individuals | R2 | $3 | £1.50 |
| Institutions | R4 | $6 | £3.00 |

**Quaestiones Informaticae is prepared for publication by SYSTEMS PUBLISHERS (PTY) LTD**

**for the Computer Society of South Africa.**

# Database Design: Choice of a Methodology

M.C.F. King, G. Naudé

National Research Institute for Mathematical Sciences, CSIR, PRETORIA, South Africa

and

S.H. von Solms

Department of Computer Science, RAU, Johannesburg, South Africa

## Summary

Aspects are discussed of the practical application of recent relational theory. The relevant definitions, extracted from various papers, have been restated using a common notation. An attempt is made to describe unambiguously two competing design procedures: synthesis and 4NF decomposition, as a basis for the discussion. It is argued that the key structure of the synthesized relations models the 'real world' naturally; that the non-loss criterion has questionable validity in practice; that 'uniqueness' is equally a problem in both procedures; and that multivalued dependencies are difficult to 'recognize'.

## 1. Introduction

In the growing flood of database papers a number of conflicting design approaches are discernible. The present paper is concerned with two of these: the method of 3NF synthesis of relations, and the method of 4NF decomposition of a 'universal' relation. Both methods set out to derive a set of suitable relation schemas, together with a set of integrity constraints (structured and other).

The earlier work on synthesis (1976) was based on 'functional dependence' constraints and the associated inference rules, involving the synthesis of 3NF relations embodying the cover of all the given functional dependencies (fd's). The method presented here does not attempt to synthesize the stronger BCNF relations: In fact some problems have no solution which is in BCNF and still embodies all the fd's.

The next phase in the development (published in 1977) was the definition of a new constraint, multi-valued dependence, and its inference rules. In terms of this a new normal form, 4NF, was defined. The definition of multi-valued dependencies was found to be equivalent to certain lossless join conditions, and this discovery led to the adoption of the lossless join condition as a criterion for a 'good' decomposition.

At this stage, the competing design procedures were synthesis and decomposition. It is still not clear whether in practice the lossless join condition is a good decomposition criterion or not.

The latest contribution to this development stream is the concept of a generalized join dependency of which a special case is a multi-valued dependency. Inference rules for the join dependency have not been derived, but an algorithm has been published that can decide whether a given fd or join dependency is implied by a given set of join dependencies and fd's. Since a lossless join condition is equivalent to the truth of a join dependency, the algorithm is a tool for finding 'good' decompositions.

Throughout the above development of concepts and theorems, semantic problems in interpreting and applying the models have presented the greatest difficulty. Whereas the formal definitions are unambiguous (although difficult to grasp), it is difficult to recognize the constraints which are true in a given real world situation. The mapping of an assertion (in narrative form) about the real world into an equivalent formal constraint often seems ambiguous.

This is probably the major difficulty in the practical context. The present paper commences by assembling the relevant definitions in Section 2. As the paper is not tutorial in nature, no attempt is made to expand on the definitions. Sections 3 and 4 describe the competing procedures, synthesis and decomposition. Section 5 discusses the procedures from various points of view, but always keeping their practical application in mind.

Finally, Section 6 attempts to resolve the practitioner's dilemma by suggesting a methodology that takes all the theory into account without becoming impractical.

It should be mentioned that there is an alternative 'thread' of papers and concepts which places little emphasis on the formal dependency constraints, or on redundancy in the formal sense. It seems that this alternative involves fewer semantic problems and may be the better approach, but space and time prevent its discussion here.

## 2. Definitions

*Attribute*

Attributes are symbols taken from a finite set U = {A1,A2, . . .}.

*Domain*

The domain DOM(Ai) is the set of possible values for the attribute Ai.

*X-value*
For the set of attributes X, an X-value is an assignment of values to the attributes of X from their domains. It is denoted by the lower case letter x.

*Relation*
A relation R on the set of attributes{A1,A2,. . . ,An} is a subset of the cross product DOM(A1)X. . . XDOM(An) denoted R(A1,A2,A3,. . ., ,An). A relation R on the union of the sets of attributes X, Y, . . . is denoted R(X,Y, . . .).

*Projection*
If u is an element of R(X) (a tuple of R(X)) and A is an element of X (an attribute in X), then u[A] is the A-component of u. If Y is a subset of X, then u[Y] is the tuple containing the Y-value from u. The projection of R on Y, R[Y], is {u[Y] | u is an element of R}. R[Y,W] denotes R[Y union W].

*Natural join*
Let R(1), R(2), . . . , R(m) be relations on the attribute sets X(1), X(2), . . . , X(m). Then the join if R(1), R(2), . . . , R(m) is defined as
JOIN(i=1,. . .,m of R(i)) = {w|condition 1 and condition 2}, where condition 1 is (w is an element of the cross product of the domains of all the attributes in the union of X(1),. . .,X(m)) and condition 2 is (for all i, 1<= i<=m, there exists a (u(i) which is a tuple of R(i) such that u(i) =w[X(i)].
The join is written R(1)*. . .*R(m).

*Functional dependence*
A functional dependence is a binary predicate, →, on 2**U, written X →Y. X→Y is true in a relation R(U) iff |R[X]| = |R[X,Y]| where X and Y are subsets of the attributes U (for any set K, |K| denotes the cardinality of K).

*Key*
Let (R(A1,A2, . . .,An) be a relation. Let X be subset of {A1,A2,. . .,An}. X is a key of R if for every attribute Ai in {A1,A2,. . .,An), X→Ai, and no subset of X has this property.

*Embodied*
A functional dependency X→A is embodied in a relation R if X is a key of R, and A is any attribute of R.

*1NF*
A relation is in first normal form (1NF) if each domain contains simple values, i.e. the domains are not themselves relation-valued.

*Prime*
If an attribute Ai appears in any key of a relation R, it is called prime in R.

*2NF*
A relation is in second normal form (2NF) if it is in 1NF and each of its non-prime attributes is fully dependent upon every key of R(X→Y is called full dependence if there is no W, a subset of X, such that W→Y).

*Transitive dependence*
Let R(A1,. . .,An) be a relation. An attribute Ai is transitively dependent upon a set of attributes X if there exists a set of attributes Y which is a subset of {A1, . . ., An} ends
That X→Y, nor Y→X, and Y→Ai,
where Ai is an element of neither X nor Y.

*3NF*
A relation is in third normal form (3NF) if none of its non-prime attnoutes are transitively dependent upon any key.

*BCNF*
A relation R is in Boyce-Codd normal form (BCNF) if the existence of an attribute A in R and a set of attributes X in R, with A not an element of X, and X→A, implies that every attribute in R is functionally dependent on X.

*R[x, Y]*
Given a relation R(U), and X, Y subsets of U, R[x,Y] = {y| for some tuple u in R, u[X] = x and u[Y] = y}, i.e. R[x,Y] is the set of all Y-values that are associated with the X-value x in R.

*Multivalued dependency*
A multivalued dependency (MVD) is a binary predicate→→ on 2**U, written X→→Y. A relation R(U) obeys the MVD X→→Y iff for every value of (X union Z), xz, R[x,z,Y] = R[x,Y] where X,Y are subsets of U, and Z is the complement of (X union Y) in U.

*Trivial MVD*
An MVD X→→Y is trivial if Z is the empty set, i.e. U = (X union Y).

*4NF*
A relation R is in fourth normal form (4NF) if whenever a non-trivial mvd X→→Y holds for R, then so does the functional dependency X→A for every attribute A of R. If a relation R is in 4NF, then it is in BCNF. If a relation R is in BCNF, then it is in 3NF.

*Proposition 1* (lossless join criterion)
Let X,Y be subsets of U, and Z be the complement of (X union Y) in U. Then X→→Y is true in R(U) iff R = R[X,Y]*R[X,Z]. Note that the equality R = R[X,Y]*R[X,Z] is equivalent to the quality R[x,Y,Z] = R[x,Y]\ R[x,Z] \ denotes the cross product for all x in R [1].

*Generalized join dependency*
The generalized join dependency (GJD) is an M-ary predicate on 2**U, written X(1)*X(2)*. . .*X(m) where X(i) is a subset of U for all i. It is true in R(U) iff R[X] = R[X(1)]*. . .*R[X(m)] where X is the union of X(i) over i = 1 to m.

*MVD versus GJD*
The multivalued dependency is a special case of a generalized join dependency, with m = 2, and X = U, i.e. it is a full binary join dependency.

*FD as a special case*
A functional dependency X→Y is a special case of a multivalued dependency with |R[x,Y]| = 1 for all x in R. If X→Y in R, then X→→Y in R.

*Span*
Suppose relation R(i) is defined on the attributes X(i). Then R(i) is said to span the functional dependency X→Y if the union of X and Y is a subset of X(i).

*Surrogate*
A name of an entity such that the (imaginary) name values are assumed to identify instances of the entity uniquely. Surrogate values are not storable. The surrogate is later replaced by suitable identifying attributes, e.g. the surrogate 'employee' may be replaced by 'emp-number, department-number' or by 'id-number'.

## 3. The Synthesis Procedure (to obtain 3NF relations)

The procedure outlined below is used iteratively. In particular, name changes in step S80 require that S70 should be repeated.
S10. All relevant attributes are named and defined.

S20. A list of functional dependencies of interest is drawn up.
S30. Note: The functional dependencies are between subsets of attributes.
S40. Note: They are chosen according to the designer's understanding of the attributes and their associations.
S50. Note: This list includes fd's with dummy right-hand sides representing the non-functional associations [2].
S60. Optionally, a functional dependence diagram is drawn to support insight, but the list remains the official record.
S70. The fd's are processed by Bernstein's Algorithm [2] to produce a list of redundant fd's, and a list of synthesized relations.
S80. By inspection, fd's which were erroneously marked redundant owing to semantic ambiguity, are identified and the appropriate attributes renamed.
S90. For each multiple key in a relation, a preferred key is chosen for use in other relations (in a non-key role), and substituted.
S99. The final set of relations is represented on a Bachman diagram.

## 4. The 4NF decomposition procedure (based on the lossless join criterion)

D1. All relevant attributes are named and defined.
D2. A single relation schema R is formed (as a concept) on all the attributes. No extension of R is given.
D3. A list is drawn up of those fd's of interest that should hold in R.
D4. A list is drawn up of those MVDs of interest that should hold in R.
D5. The following algorithm is executed. (Where the attributes of a relation S are denoted by A(S).)

```
Rels: = {R: (Rels is a set of relations);
Repeat;
   for each relation S in Rels;
   if there exist X,Y which are subsets of A(S);
   and Z, the complement of (X union Y) in A(S)
   is not empty; and $X \longrightarrow Y$, and not $X \rightarrow Z$;
   then Rels: = (Rels – {S})union{S[X,Y],S[X,Z]};
Until all relations in the set Rels are in 4NF.
```

The algorithm uses the results:

```
   if $X \twoheadrightarrow Y$ in R(X,Y,Z), then
   $X \twoheadrightarrow Y'$ in R[X,Y',Z'] where Y' is a subset of Y,
   and Z' is a subset of Z;
   if $X \rightarrow Y$ in R(X,Y,Z), then
   if $X \rightarrow Y'$ in R[X,Y',Z'] where Y' is a subset of Y,
   and Z' is a subset of Z.
   If $X \rightarrow Y$ in R, then $X \twoheadrightarrow Y$ in R.
```

D6. Every relation is now in 4NF, and therefore in BCNF and 3NF, but in general there may be some fd's not embodied in the relations. With each relation, list all constraints that it must satisfy (there will be no nontrivial MVDs that are not fd's), and use them to identify its one or more keys.
List the fd's not embodied in the relations. They must be explicitly enforced on the database.

## 5. The Procedure Discussed from Different Viewpoints

### 5.1 Representing the 'real world'

It is assumed that the real world consists of concepts such as: entities (such as employee, project, . . .);
single-valued properties of entities (such as the current surname of an employee);
multiple-valued properties of entities (such as the children of an employee);
0/1 properties of entities (such as 'reason for firing');
associations between entities (such as an 'employee, project' association);
events (such as death, birth, payment of account, order of spares).
The associations between entities are themselves entities (abstract) and may have properties. For instance, the entity 'employee, project' has the property 'hours worked to date'.
There is an important correspondence between these informal concepts and the formal concepts of relational theory. By the definition of a 'key', every relation in 1NF has at least one key. If a 1NF relation has more than one key, then the keys are equivalent (there is a bijection between them) whether or not the relation is in 3NF. Each 1NF relation may therefore be interpreted to represent exactly one entity, viz the entity whose instances are uniquely identified by the key values. Call this the relation-entity.
If a relation is in 3NF then its non-prime attributes, if any, are direct properties of the relation-entity; whereas if a non-3NF relation has non-prime attributes, they may be indirect properties of the relation-entity, or properties of some other entity.
For instance, suppose
  EMPLOYEE → NAME
  EMPLOYEE → DEPARTMENT
  STOCK-NUMBER, DEPARTMENT → QUANTITY
  DEPARTMENT → MANAGER.
In the relation R(STOCK-NUMBER, DEPARTMENT, QUANTITY, MANAGER) the relation-entity is the association between stock-items and departments.
Whereas QUANTITY is a direct property of this association, MANAGER is a property of some other entity, viz department. R is not in 3NF.
The relation S(EMPLOYEE, NAME, DEPARTMENT, MANAGER) has the associated relation-entity, employee. MANAGER is a property of DEPARTMENT, which is a property of employee, i.e. MANAGER is an indirect property of employee. S is not in 3NF.
The method of synthesis always creates a minimal set of 3NF relations embodying a cover of the given fd's. The following correspondence is therefore guaranteed.
1. Every relation in the solution represents one entity and contains only direct properties of that entity.
2. Each entity is represented by one relation.
It is the key structure of the 3NF relations which 'captures' the 'entity, property' structure of the real world (a part of the real semantics).
The decomposition method is less satisfying. Many of the alternative 4NF solutions to a problem do not embody all the functional dependencies. Since the relations are in 3NF (4NF $\Rightarrow$ 3NF), each relation represents an entity with its direct properties; however, the entities sometimes represent 'peculiar' associations: for instance in the example of Fagin [3] shown in 5b, a different decomposition can be found which contains the relation R221(CLASS-SECTION, RANK, SALARY), in which the rank and salary are actually those of an instructor who teaches the class and section.
Some problems have no 4NF solutions which embody all the fd's.

## 5.2 The significance of the loss-less join criterion

Why use the loss-less criterion as a basis for finding a decomposition into base relations? The following reasons are usually given or implied.
1. The initial single relation may be recovered without loss of information.
2. The initial single relation satisfies all constraints. If it can always be recovered without loss of information, then in some sense the constraints are 'preserved'.
However a number of 'weaknesses' in these reasons may be noted.
The loss-less criterion involves the operations of projection and join, which are defined in terms of the extension of a relation (its tuples),

not just its schema, yet in the application of the criterion only the schemas are actually involved.

If R has a non-loss decomposition R(1),. . .,R(m) on the attribute sets X(1),. . .,X(m), this means that given an extension of R (the actual tuples), R may be projected onto X(1),. . .,X(m) respectively to give R(1),. . .,R(m), and provided that R(1) to R(m) are not updated after projection, the original R may be recovered by joining them.

If however the relations R(i) are updated before being joined, then clearly the original R is not recovered. In this case the constructed relation might not even satisfy the same constraints.

In the first case, R(i) = R[X(i)]; while in the second case, R(i) = modified-R[X(i)].

If, as in the 4NF algorithm, the decomposition procedure involves only the relation schemas and not their extensions, then there is not even an extension of R to project, let alone to recover. In this case the 'projections' are actually relation schemas having no initial extensions. It is the intent of the 4NF decomposition method that these 'base' relations will constitute the logical view of the database. The 'full' join of all base relations will probably never be referenced during manipulation of the database. The update and retrieval activities will each be confined to the join of the smallest subset of base relations that satisfies the information requirements of retrieval, or the integrity needs of update.

Thus, not only is there no single relation extension to start with, but also the projections (base relations) are being continuously updated. What then is the value of a 'loss-less decomposition' choice of the base relations? Since we neither start with nor ever actually recover a single relation extension, the loss-less criterion must be seen as a means of finding a set of base relations which are non-redundant, and which in some sense ensure that the integrity constraints are always obeyed.

The argument seems to be that although the single relation is never materialized as an extension, any hypothetical instance of it obeys all constraints. Therefore when this instance is projected onto the loss-less components, even though some constraints may not be spanned or embodied by any component or relation, all constraints will be true in the full join. It seems to be assumed that to any set of component relation extensions, there corresponds an instance of the single relation satisfying all the constraints.

As far as actual data (the extensions) is concerned, the starting point is the base relations (they are the first to be populated with data). If an instance of a single relation is ever materialized, it will be by joining all the base relations. That is, the join operation will not be preceded by projection. From this viewpoint, the emphasis moves from the inclusion relation, viz 'R is a subset of R[X(1)]*. . .*R[X(m)]', which underlies the non-loss criterion, to the inclusion relation 'R(i) is a subset of (R(1) . . . R(m))[X(i)] for i = 1,. . .,m'. This last inclusion casts doubt on whether an arbitrary 'real world' situation can always be represented by a single relation. For instance, in the 'valid code' case, where (job) codes currently used bv employees are a subset of allowed (job) codes, the single relation R1(EMPLOYEE, CODE, DESCRIPTION) would lose some valid codes. The relation R2(EMPLOYEE, USED-CODE, VALID-CODE, DESCRIPTION), in which the CODE attribute has been renamed, represents the situation without loss of information (loss associated with the second inclusion relation above).

Even if it is always possible to 'capture' the reality in a single relation, it will in general be necessary to rename attributes so that

R(i) = (R(i)*. . .*R(m))[X(i)] for all i.

Moreover, once the decomposition has been made, the attribute names will have to be changed back if, for instance, it is required to list the values of EMPLOYEE, CODE and DESCRIPTION, since a meaningful join will be obtained only if the code attribute has the same name in the two relations. The descriptions of only the used codes will be listed. If the names are left as VALID-CODE, and USED-CODE, the join will yield the Cartesian product rather than the descriptions of codes used by employees.

A case where the integrity of the full join is not preserved is illustrated by a 'real life' example dealing with stands and the firms which occupy them. Groups of firms occupy groups of stands in such a way that particular stands are not associated with particular firms, but rather, partitions are defined, each of which contains a group of stands and a group of firms. The fd's in the single initial relation R(STAND,FIRM,PARTITION), are STAND → PARTITION and FIRM →PARTITION and the MVD PARTITION →→STAND. A possible loss-less decomposition is therefore R1(STAND,PARTITION), R2(STAND,FIRM), in which the second fd is not spanned. Joining R1 and R2 after independently updating them will therefore not preserve integrity.

It can also be shown that even when every fd is embodied in some 'loss-less' component relation, and consequently the full join obeys all constraints, a non-full join, even though it spans all the attributes, does not necessarily obey all the constraints.

This is illustrated by an example from Fagin [3], with the following constraints:

CLASS-SECTION →CLASS, INSTRUCTOR
CLASS-SECTION, DAY →ROOM
STUDENT →MAJOR, YEAR
INSTRUCTOR →RANK, SALARY
CLASS-SECTION →→STUDENT, MAJOR, SCORE, YEAR
CLASS-SECTION →→INSTRUCTOR, RANK, SALARY
CLASS-SECTION →→TEXT
CLASS-SECTION →→DAY, ROOM
CLASS →→TEXT
CLASS-SECTION, STUDENT →→SCORE.

Starting with a single relation on all the attributes, a possible 4NF decomposition is:

R11 (CLASS-SECTION, STUDENT, SCORE)
R121 (STUDENT, MAJOR, YEAR)
R122 (STUDENT, CLASS-SECTION)
R211 (INSTRUCTOR, RANK, SALARY)
R212 (INSTRUCTOR, CLASS-SECTION)
R221 (CLASS-SECTION, TEXT)
R2221 (CLASS-SECTION, CLASS)
R2222 (CLASS-SECTION, DAY, ROOM).

A non-full join of all relations except R212 is a relation spanning all the attributes, yet it does not satisfy the first fd.

Thus not only is loss-less decomposition on its own insufficient to guarantee the integrity of the (never to be materialized) full join, but even when the full join does obey all constraints, the non-full joins do not necessarily do so.

Yet, as was stressed previously, the modus operandi of the database system relies heavily on non-full joins, and seldom if at all on the full join.

Whether synthesis or 4NF decomposition is used, a set of base relations is the result. In both update and retrieval it is sometimes sufficient to access one base relation, and at other times necessary to access the join of multiple relations. It seems reasonable to require that this access relation, whether a single base relation or a join, should satisfy all of the fd's (given as constraints) which it spans. The non-loss criterion on its own does not appear to enforce this condition (Rissanen's independence criterion addresses this lack [6]).

## 5.3 Uniqueness

In both the method of synthesis and that of decomposition the assumption of uniqueness has to be made although it takes different forms. In the case of synthesis, any two fd's with the same left-hand and right-hand sides are 'the same fd'. This concept differs from that of a function in mathematics, where differently named functions may be defined on the same domain and range ($y = f(x), y = g(x)$), and composition may differ significantly, e.g. $z = h(y)$ can lead to $z = h(f(x))$ or $z = h(g(x))$.

(In the DIAM binary model the access language makes use of named paths, and seems to avoid semantic ambiguity of the type experienced with synthesis.)

In the 4NF decomposition method it is implicit that no two columns of the single relation may have the same attribute names. This is equivalent to the synthesis uniqueness assumption.

The uniqueness assumption is associated with some problems of semantic ambiguity that frequently arise in practice. These problems are illustrated by some synthesis examples from Bernstein [2].

Let

f1: DEPARTMENT →MANAGER

f2: MANAGER,FLOOR → NUMBER-OF-EMPLOYEES

f3: DEPARTMENT,FLOOR → NUMBER-OF-EMPLOYEES

and suppose that a manager manages more than one department

Then g3: DEPARTMENT,FLOOR → NUMBER-OF-EMPLOYEES can be derived from f1 and f2 using pseudo-transitivity. The meaning of g3 (by inspection of its derivation) is the number of employees of the manager of the department on the given floor, whereas f3 means 'the number of employees of the department on the given floor'.

Since g3 is syntactically identical to f3, f3 will be wrongly erased in the algorithm of Bernstein.

It should therefore be standard procedure to examine all the redundant fd's (which are listed by algorithm 2).

Semantic ambiguity in these fd's can be detected with the aid of the data dictionary. For instance a typical entry might be 'NUMBER-OF-EMPLOYEES: The number of employees of a manager on a particular floor'.

In the case of fd's which have been wrongly erased, the relevant attributes are then renamed to remove ambiguity. In the above case, the renaming gives

f2: MANAGER, FLOOR →EMPLOYEES-OF-MNG

f3: DEPARTMENT, FLOOR →EMPLOYEES-OF-DEPT.

Then g3 becomes

DEPARTMENT, FLOOR →EMPLOYEES-OF-MNG

It is still possible that semantically different but syntactically identical fd's may exist in the closure. For example,

A →B,K

B →C

K →C

in the cover, leads to A →B →C and A →K →C in the closure. This is not well understood but it seems that careful interpretation of the joins that are actually used would reveal their correct meaning.It is claimed for 4NF decomposition that starting with a single relation makes semantic ambiguity visible from the beginning. In the previous example, the single relation would be on the attributes

(DEPARTMENT,MANAGER,FLOOR, NUMBER-OF-EMPLOYEES).

As before, the data dictionary would have an entry for NUMBER-OF-EMPLOYEES, and in any case it is claimed that inspection of the single relation would reveal the ambiguity in f2 and f3. Conceptually the ambiguity is indeed easier to notice in the sense that an attribute can have only one meaning in a single relation.

However, when it is realized that in a 'real' problem there may be hundreds of attributes and that both synthesis and decomposition essentially start off with a given set of attributes (whether regarded as a single relation or not) and a set of constraints, then it is not clear that 4NF decomposition has any advantage in revealing ambiguity.

A good illustration is given below of the conceptual clarity that a single relation view can provide.

When listing a set of fd's without regard to a 'universal' containing relation, it was thought that

Employee-number → Manager (the manager of the employee)

Manager → Employee-number-of-manager (a manager has his own employee number)

and 'of course' every employee-number-of-manager is an employee-number therefore

Employee-number-of-manager → employee-number.

When the attributes were seen as belonging to a single relation it was clear that the last fd could not coexist in the relation without destroying the meaning that a manager has many employees. However, using synthesis, the error can be as easily discovered by examining the bijections.

## 5.4 Recognizing MVDs

Given a single relation schema on a set of attributes, but no instance of the relation, how does one set about recognizing the MVDs that are to be adopted as constraints which the relation must satisfy? Since the formal definition of an MVD is unambiguous, if an instance of the relation were given, all MVDs holding in it could be recognized. However, the decomposition method starts with a single relation schema, not an extension. The required MVDs must therefore be identified solely on the basis of the analyst's understanding of the attributes and their associations.

This understanding is recorded in narrative form. The question may then be rephrased as 'how does one recognize the necessary MVDs from an inspection of the narrative only?'.

Interesting observations were made when the narrative of the example due to Fagin, described in 5b, was presented independently to two colleagues. Colleague A works in the area of category theory in mathematics and had for about 6 months been exposed in depth to the notion of MVDs. Colleague B has an honours degree in Computer Science and had been exposed to MVDs only briefly but also in some depth.

Colleague A correctly identified all the fd's and some of the MVDs, but wrongly identified the following MVDs

CLASS-SECTION →→STUDENT

CLASS-SECTION →→DAY,

and omitted the MVDs

CLASS-SECTION →→ STUDENT,MAJOR,SCORE,YEAR

CLASS-SECTION →→ INSTRUCTOR,RANK,SALARY

CLASS-SECTION →→ DAY,ROOM.

Colleague B arrived at exactly the same wrong result.

The derivation of the first incorrect MVD above was of particular interest. B was looking for subsets of attributes X, Y and their complement Z, such that 'a set of Y-values is associated with an X-value and is independent of the Z-values'. It appeared to B that STUDENT was independent of MAJOR and YEAR. He could see that MAJOR and YEAR were dependent on STUDENT but not vice versa.

The suggested ways of recognizing an MVD are enumerated below.

1. In the relation R(EMPLOYEE, CHILD, SALARY), EMPLOYEE $\twoheadrightarrow$ CHILD holds for R because intuitively an employee's set of children is completely determined by the employee and is 'orthogonal' to the salary.

An appreciation of the formal properties of MVDs may be of assistance in recognizing them even in the absence of an extension.

2. $X \twoheadrightarrow Y$ in R(X,Y,Z) iff whenever (x,y,z) and (x,y',z') are tuples of R so are (x,y,z') and (x,y',z).
3. $X \twoheadrightarrow Y$ holds for R(X,Y,Z) iff Y and Z are 'orthogonal' or 'independent' sets of column names.
   In this respect note the case
   {empty} $\twoheadrightarrow$ Y and
   {empty} $\twoheadrightarrow$ Z where X = { }
   which is illustrated by a 'single relation' representation of the traditional 'Bill of Materials' problem, viz
   { } $\twoheadrightarrow$ PART-NUMBER, DESCRIPTION
   { } $\twoheadrightarrow$ ASSEMBLY-NUMBER,COMPONENT-NUMBER,QUANTITY.
4. Noting that $X \twoheadrightarrow Y$ in R(X,Y,Z) implies $X \twoheadrightarrow Z$ in R, the intuitive meaning of $X \twoheadrightarrow Y$ is that there are really two independent relation schemas R1(X,Y) and R2(X,Z).

Some properties of MVDs that complicate their recognition are shown below.

1. Although $X \twoheadrightarrow$ A,B in R, $X \twoheadrightarrow$ A is not necessarily true.
2. Although $X \twoheadrightarrow Y$ is true in R1 (say) it is not necessarily true in R1*R2. Although $X \twoheadrightarrow Y$ is false in R (say) it may be true in a projection of R.
3. An MVD $X \twoheadrightarrow Y$ in R does not correspond to the simple concept 'with each X-value there is an associated set of Y-values' (which is trivially true in R).

Fagin's example illustrates one aspect of translating narrative to formal constraints. The narrative 'each CLASS has a set of TEXTs which are used by all SECTIONs of the class', has to be recognized as equivalent to 'the set of TEXTs are determined only by the CLASS and are orthogonal to the SECTIONs'. Alternatively the first assertion may be immediately formalized as R[class,TEXT] = R[class,class-section,TEXT], for all (class,class-section) values in R.

According to the definition of an MVD, this might then be interpreted as CLASS $\twoheadrightarrow$ TEXT is true in R, but note that CLASS-SECTION is not the complement of CLASS,TEXT in R. Thus the MVD is true only in the projection R[CLASS,TEXT,CLASS-SECTION]. We may accept the MVD as a constraint on R if we assume that TEXT is 'orthogonal' to every attribute of R except CLASS. What would this mean, considering that INSTRUCTORS are associated with TEXTs and that every instructor is not associated with every text? In this case we realize that every instructor is associated with every text of every class he teaches. This means that instructors and texts are indeed orthogonal within their class groups.

The above is typical of the cumbersome type of analysis one is forced to make.

This section has illustrated that although the decision as to whether an MVD holds in a given extension is unambiguously made from the formal definition, it is nevertheless difficult to decide from the narrative which MVDs 'should' hold in a given relation schema.

## 6. A Suggested Methodology

A factor contributing to the difficulty is that the full relation is being 'searched' for MVDs: the 'orthogonality' of the whole complement has to be established each time. If the MVD concept and its associated non-loss criterion could be used in a smaller relation context, the problem of identifying the MVDs should be far easier. Hence it is suggested that 4NF decomposition be used as an element in the synthesis procedure after the base relations have been synthesized. The intent would be to recognize relations of the type illustrated by Fagin [3], in which the key consists of all the attributes and non-trivial MVDs are present. These relations would then be suitably decomposed.

The relations synthesized by Bernstein's algorithm have their origin either in the dummy fd's used to represent nonfunctional associations, in which case the single key includes all the attributes, or in the 'true' fd's, in which case there may be more than one key, none of which includes all the attributes.

A characteristic of the synthesized relations is that they embody all the fd's. In using 4NF decomposition as a final step applied to each base relation obtained by synthesis, we shall arbitrarily ensure that all fd's remain embodied. This means that only those relations synthesized from the non-functional associations can be candidates for decomposition. This follows from the following proposition.

*Proposition*

If a 3NF relation embodying at least one non-trivial fd is not in 4NF, then the fd's embodied in the relation are not all embodied in the associated non-loss decomposition.

*Proof:*

Given R(U) is in 3NF and embodies at least one non-trivial fd. Let its keys be X1, X2, . . ., Xm and its non-prime attributes (there may be none) be B1, . . ., Bn.

Given $X \twoheadrightarrow Y$ in R(U) and there exists A an element of U such that not ($X \rightarrow A$), and Z = U – (X union Y) not = {empty} (i.e. non-trivial MVD),

then R = R[X,Y]*R[X,Z].

Suppose the set of fd's embodied in R is equal to the union of the sets of fd's embodied in R[X,Y] and R[X,Z].

Then at least one key of R, Xr say, is a key of both R[X,Y] and R[X,Z]; (for if no Xr were a key of both projections, then all keys of R would have to be keys of one projection – else bijections would be lost – and all non-prime attributes of R would have to be attributes of the same projection, which contradicts Z not = {empty}).

Therefore Xr is a subset of X, and $X \rightarrow$ Xr. It follows that $X \rightarrow$ {all attributes of R}. This contradicts the given condition (not $X \rightarrow$ A). Therefore the fd's embodied in R are no longer embodied in the projections.

End of proof.

*A modified synthesis procedure*

Only the steps to be added to the original synthesis procedure are shown.

S1. Every entity of interest is named (using surrogates) and defined.

S11. Assertions (in natural language) about the entities and attributes, are noted and agreed upon with the 'user'.

S31. Note: The fd's are also between entities and entities, and between entities and attributes.

S41. Note: They are chosen also according to the designer's understanding of the entities.

S81. If any bijection (multiple keys in a relation) does not make sense, the offending fd is deleted, or the relevant attribute renamed.

S82. For each synthesized relation, the MVDs of interest are noted (the assertions being kept up to date in the process).

S83. Suppose S is the set of synthesized relations. While S contains a relation T(X) which is not in 4NF and whose key is X (i.e. a relation induced by a non-functional association), T(X) is replaced by its components.
S84. For the remaining non-4NF relations the MVD constrains to be explicitly enforced are noted. The relations are not decomposed.
S85. Suitable attributes are chosen which uniquely identify each entity, and these attributes substituted for the surrogates.
Thus, in this modified synthesis procedure, MVDs will have two effects. Some relations, having their origin in nonfunctional associations, will be decomposed, while some MVDs will have to be explicitly enforced.

## References

[1] BEERI, FAGIN and HOWARD. (1977). A complete axiomatization for functional and multivalued dependencies in database relations. *Proc. ACM SIGMOD.* Int. Conf. on Manag, Toronto, Canada, 47-61.

[2] BERNSTEIN, P.A. (1976). Synthesising third normal form relations from functional dependencies. *ACM TODS,* **1,4** 272-298.

[3] FAGIN, R. (1977). Multivalued dependencies and a new normal form for relational databases. *ACM TODS,* **2,** 3, 262-278.

[4] FAGIN, R. (1977). Decomposition versus the synthetic approach to relational database design. Proceedings 1977 Very Large Database Conference (Tokyo), Third Int. Conf. on Very Large Data Bases, Tokyo, Oct. 1977, 441-446.

[5] KING, M.C.F., NAUDE, G. and VON SOLMS, S.H. (1979). NRIMS Note (to be published).

[6] RISSANEN, J. (1977). Independent components of relations. *ACM TODS,* **2,** 4, 317-325.

# Notes for Contributors

The purpose of this Journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles, exploratory articles or articles of general interest to readers of the Journal. The preferred languages of the Journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be in double-spaced typing on one side only of A 4 paper and submitted to Dr. D. S. Henderson or Prof. M. H. Williams at

Rhodes University
Grahamstown 6140
South Africa

**Form of manuscript**

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. The original ribbon copy of the typed manuscript should be submitted. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name, and the affiliation and address. Each paper must be accompanied by a summary of less than 200 words which will be printed immediately below the title at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

**Tables and figures**

Illustrations and tables should not be included in the text, although the author should indicate the desired location of each in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Illustrations should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the Author's name and figure number. Original line drawings (not photoprints) should be submitted and should include all relevant details. Drawings, etc., should be about twice the final size required and lettering must be clear and "open" and sufficiently large to permit the necessary reduction of size in block-making.

Where photographs are submitted, glossy bromide prints are required. If words or numbers are to appear on a photograph, two prints should be sent, the lettering being clearly indicated on one print only. Computer programs or output should be given on clear original printouts and preferably not on lined paper so that they can be reproduced photographically.

Figure legends should be typed on a separate sheet and placed at the end of the manuscript.

**Symbols**

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters between the letter O and zero; between the letter l, the number one and prime; between K and kappa.

**References**

References should be listed at the end of the manuscript in alphabetical order of author's name, and cited in the text by number in square brackets. Journal references should be arranged thus:

1. ASHCROFT, E. and MANNA, Z. (1972). The Translation of 'GOTO' Programs to 'WHILE' Programs, in *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255.
2. BÖHM, C. and JACOPINI, G. (1966). Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM,* **9,** 366-371.
3. GINSBURG, S. (1966). *Mathematical Theory of Context-free Languages,* McGraw Hill, New York.

**Proofs and reprints**

Galley proofs will be sent to the author to ensure that the papers have been correctly set up in type and not for the addition of new material or amendment of texts. Excessive alterations may have to be disallowed or the cost charged against the author. Corrected galley proofs, together with the original typescript, must be returned to the editor within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Fifty reprints of each article will be supplied free of charge. Additional copies may be purchased on a reprint order form which will accompany the proofs.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

**Letters**

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.

Hierdie notas is ook in Afrikaans verkrygbaar.

# Questiones Informaticae

Partial proceedings of the first South African Computer Symposium on Research in Theory, Software, Hardware, organised by The Research Symposium Organising Committee of The Computer Society of South Africa. 4 & 5 September 1979, Pretoria.

## Contents/Inhoud