NEST AND UNNEST OPERATORS IN NESTED RELATIONS

Georgia Garani

Dept of Computer Science and Telecommunications, Technological Educational Institute, Larissa 41110, Greece
Email: garani@teilar.gr

ABSTRACT

By distinguishing nested attributes as Decomposable and Non-Decomposable, it is proved that for all nested relations, unnesting and then renesting on the same attribute yields the original relation subject only to the elimination of duplicate data. Therefore, the statement that was popular in nested relations research: "Unnesting and then nesting on the same attribute of a nested relation does not always yield the original relation" is reconsidered.

Keywords: Non-First Normal Form database, Nested relation, Decomposable nested attribute

1 INTRODUCTION

The Nested or Non-First Normal Form (N1NF) data model was first introduced by Makinouchi (1997), in an effort to relax the first normal form (1NF) assumption, i.e. attributes must contain only atomic values. In contrast, in the N1NF data model, relations are allowed to have both atomic and non-atomic attributes that is the value of an attribute can be a relation itself, which can be considered as a subrelation of the relation. The N1NF data model allows each such subrelation to have relation-valued attributes, thus allowing nesting of relations or embedding of one relation within another. In this paper, relations that contain relation-valued attributes are called *nested relations*, whereas relations that do not contain relation valued attributes are called *flat relations*.

The N1NF data model has been widely investigated since 1977, as it has offered good solutions for a number of applications, such as engineering design and textual data (Hernández, 1993). The major advantage claimed for nested relations over flat relations is that they minimise data fragmentation (Garani, 2006). The nested structure of the relation makes the semantics of the stored data clear. In addition, redundancy and update anomalies can be minimised and less storage is needed as one relation in non-first normal form can represent many relations in first normal form representation (Gyssens, Suciu & Van Gucht, 2001; Liu, Vincent & Mohania, 1999). As a result, query processing can be faster because a smaller number of join operations may be required (Despande & Larson, 1991).

This paper is structured as follows. In Section 2 definitions of nested relations are given and two different categories of nested attributes are distinguished based on the semantics of the data they hold. In Section 3, this distinction is used to demonstrate that unnesting and then nesting on the same attribute of a nested relation R returns the original relation R, subject to the elimination of duplicate tuples. Therefore, the proposition that unnesting a nested relation R and then nesting it on the same attribute does not, necessarily, give the original R relation at all times, which is mentioned in Fischer & Thomas (1983), Jaeschke & Schek (1982), Schek & Scholl (1986) and Thomas & Fischer (1986), is reconsidered. It is shown that this is only a theoretical issue because a case such as that discussed in the earlier papers either cannot exist in the real world or if it does, that no information is lost by unnesting a nested relation and then nesting it on the same attribute. Finally, Section 4 concludes the paper.

2 DEFINITIONS OF NESTED RELATIONS

Nested relations are relations that contain one or more nested attributes. Nested attributes are the attributes which have non-atomic values, i.e., they have relations as values, which can have one or more nested attributes and so forth. The nested attributes are in fact subrelations of the nested relation to which they belong.

Definition 1 (*Nested Relation Schema*) The schema of a relation r in the N1NF data model is defined recursively as RS = $R(R_1S_1, R_2S_2, ..., R_nS_n)$, where $n \geq 1$, $R_1, R_2, ..., R_n$ are the attribute names of R, either atomic or nested and

$$S_i = \begin{cases} \varnothing \text{ (empty set)} & \text{if } R_i \text{ is an atomic attribute} \\ (R_{i1}S_{i1}, R_{i2}S_{i2}, ..., R_{ik}S_{ik}) & \text{if } R_i \text{ is a nested attribute and } k \geq 1 \end{cases}$$

where $1 \leq i \leq n$.

An attribute or set of attributes whose values uniquely identify each entity in an entity set is called a *key* for that entity set (Ullman, 1988). For the case of a nested database model, entity sets are nested relations, and the definition of the key must be expanded in order to support nested attributes as well. Informally, a nested relation can have either atomic or nested attributes or even a combination of atomic and nested attributes as a key. Semantically, a nested attribute is a key of a nested relation, when each set of values of the nested attribute that belongs to the same tuple uniquely identifies that tuple. That implies that each of these sets of values of the nested attribute distinguishes, as an entirety, solely the tuple in which it belongs.

Definition 2 (*Key of a nested relation*) The key of a nested relation schema R can be a set K, consisting of atomic and/or nested attributes of R, such that for any two tuples t_i and t_j in the relation, the following constraint is valid at all times: $t_i[K] \neq t_j[K]$, where $i \neq j$ and with the additional property that removing any attribute from K leaves a set of attributes that is not a key of R.

Definition 3 (*Two tuples agree on an attribute*) Let R be a nested relation schema, and t_1 and t_2 be two tuples from R. Let also A be an attribute either atomic or nested, $Attr(R_a)$ be all the atomic attributes, and $Attr(R_n)$ all the nested attributes of R. The two tuples, t_1 and t_2, agree on A if the followings hold:

- If $A \in Attr(R_a)$, then $t_1[A] = t_2[A]$
- If $A \in Attr(R_n)$, then for $\forall\ t'_1 \in t_1[A], \exists\ t'_2 \in t_2[A]$ such that t'_1 and t'_2 agree on all attributes of A $\wedge$ for $\forall\ t'_2 \in t_2[A], \exists\ t'_1 \in t_1[A]$ such that t'_2 and t'_1 agree on all attributes of A.

Definition 4 (*Functional Dependency*) Let R be a nested relation schema and A and B be two sets consisting of atomic and/or nested attributes of R. B is functionally dependent on A, or A functionally determines B, denoted as $A \rightarrow B$, if whenever two tuples of R agree on all attributes in A, they also agree on all attributes in B.

Functional dependency is a constraint determined by the semantics of the relation. That is, functional dependency is not a property derived from a specific relation. The above definition allows either the left hand side or the right hand side of a functional dependency to be a relation.

Two new terms are defined below, called Nested Attribute Set and Nested Attribute Element. Informally, the set of values of a nested attribute in a tuple in a nested relation is called *Nested Attribute Set*. A Nested Attribute Set consists of a finite number of tuples since it is a relation (subrelation) itself. Each tuple of this subrelation is called a *Nested Attribute Element.*

Definition 5 (*Nested Attribute Set*) Let $\{A_1, A_2, ..., A_n\}$ where $n \geq 1$, be a set of attribute names and let $R = (A_1, A_2, ..., A_n)$ be the name of the schema of a nested relation with an instance r. Without loss of generality, suppose A_i, where $1 \leq i \leq n$, is a nested attribute and t is a tuple of r, i.e. $t \in r$. Then $t[A_i]$ is called Nested Attribute Set.

Definition 6 (*Nested Attribute Element*) Let $t[A_i]$ be a Nested Attribute Set and let $A_i = (B_1, B_2, ..., B_m)$, where $m \geq 1$, be a subrelation schema, and B_j is the name of an atomic or nested attribute, where $1 \leq j \leq m$. Then, $t[A_i] = \{t_1[B_1], t_1[B_2], ..., t_1[B_m], t_2[B_1], t_2[B_2],..., t_2[B_m], ..., t_q[B_1], t_q[B_2], ..., t_q[B_m]\}$ where $q \geq 1$. A Nested Attribute Element is a member of the set of $t[A_i]$, i.e. is a tuple $t_k[B_1]\ t_k[B_2], ..., t_k[B_m]$ where $1 \leq k \leq q$.

The power set of all the possible Nested Attribute Elements of the attribute A is the *domain* of this nested attribute.

Example 1.
In Figure 1, the set {a1, a3} is a Nested Attribute Set of the nested attribute A whereas a1 is a Nested Attribute Element of the nested attribute A. Similarly, the set {a3, a4, a5} is a Nested Attribute Set of A whereas a5 is a Nested Attribute Element of A.

A	B
A′	
a1 a3	b2
a3 a4 a5	b1
a2	b3

Figure 1. The nested relation R

According to the above definitions, in Figure 1, R = (A, B) = (A(A′), B), t[R] = ({a1, a3}, b2), ({a3, a4, a5}, b1), ({a2}, b3), t[A] = {a1, a3}, {a3, a4, a5}, {a2}.

Nest operation compares component values using a certain equality function and groups the values into sets with nesting components. Unnest operation is the inverse of Nest. It is used to restructure groups of sets eliminating a level of brackets and then eliminating duplicates (Thalheim, 2000). The formal definitions of Nest and Unnest operations (Thomas & Fischer, 1986) are given below for completeness.

Definition 7 (*Nest Operation*)
Let R be a relation scheme in database scheme S. Let r be an instance of relation scheme R. Let $\{B_1, B_2, \ldots, B_m\} \subset E_R$ (E_R is the set containing names on the right-hand side of R) and $\{C_1, C_2, \ldots, C_k\} = E_R - \{B_1, B_2, \ldots, B_m\}$. Assume that either the rule $B = (B_1, B_2, \ldots, B_m)$ is in S or that B does not appear on the left-hand side of any rule in S and $(B_1, B_2, \ldots, B_m)$ does not appear on the right-hand side of any rule in S. Then, $\mathrm{NEST}_{B = (B1, B2, \ldots, Bm)}$ (<R, r>) = <R', r'>, where

1. the rules $R' = (C_1, C_2, \ldots, C_k, B)$ and $B = (B_1, B_2, \ldots, B_m)$ are appended to the set of rules in S if they are not already in S,
2. r' = {t | there exists a tuple $u \in r$ such that
   $t[C_1, C_2, \ldots, C_k] = u[C_1, C_2, \ldots, C_k] \wedge t[B]$
   $= \{v[B_1, B_2, \ldots, B_m] \mid v \in r_1 \wedge v[C_1, C_2, \ldots, C_k] = t[C_1, C_2, \ldots, C_k]\}\}$.

Definition 8 (*Unnest Operation*)
Let R' be a relation scheme in database scheme S. Assume B is some higher-order name in $E_{R'}$. This means there must be a rule $B = (B_1, B_2, \ldots, B_m)$ in S. Let $\{C_1, C_2, \ldots, C_k\} = E_{R'} - B$.
Then, $\mathrm{UNNEST}_{B = (B1, B2, \ldots, Bm)}$ (<R', r'>) = <R, r>, where

1. $R = (C_1, C_2, \ldots, C_k, B_1, B_2, \ldots, B_m)$ is appended to the set of rules in S if it is not already in S,
2. r = {t | there exists a tuple $u \in r'$ such that
   $t[C_1, C_2, \ldots, C_k] = u[C_1, C_2, \ldots, C_k] \wedge t[B_1, B_2, \ldots, B_m] \in u[B]\}$.

Nested attributes can be divided into two different categories depending on their semantics.

2.1 Definitions of the semantic categories of nested attributes

Decomposable Nested attributes consist of Nested Attribute Elements, which are logically independent of each other but form Decomposable Nested Attribute Sets only because they have one or more real world properties in common and hence, share the same value for one or more attributes. Decomposable Nested attribute(s) can, but need not, be part of the key of the nested relation.

Definition 9 (*Decomposable Nested attribute*) Let R be a nested relation. Assume A is a nested attribute of R consisting of attributes $A_1, A_2, \ldots, A_n$ ($n \geq 1$) and B an atomic attribute of R. Let also, A functionally determines attribute B (i.e., $A \rightarrow B$). A is a Decomposable Nested attribute of R iff $\mathrm{UNNEST}(_{A(A_1, A_2, \ldots, A_n)}R)$, then $A_1, A_2, \ldots, A_n \rightarrow B$.

Example 2.
SCHOOL relation in Figure 2 has two attributes: the Decomposable Nested attribute MODULE and the atomic attribute COURSE. The semantics of this relation is that in a school, a set of modules forms a course, i.e. the Decomposable Nested attribute MODULE is the key of the SCHOOL relation and therefore, MODULE → COURSE. In this case, if $\mathrm{UNNEST}(_{\mathrm{MODULE(MODULE')}}\mathrm{SCHOOL})$, then MODULE' → COURSE. Consequently, the Decomposable Nested Attribute Elements are logically independent of each other and, in general, unordered.

MODULE	**COURSE**
MODULE′	
Pascal	Programming
C++	
Word 97	Word processor
Word Perfect	
Microsoft Works	
Access	Database and Spreadsheet
Excel	
Java	Programming
Visual Basic	
Pascal	

Figure 2. The nested relation SCHOOL

If SCHOOL relation is unnested, the result is the relation in Figure 3. By unnesting it, the MODULE' attribute functional determines the COURSE attribute. Non-Decomposable Nested attributes consist of groups of values that form Non-Decomposable Nested Attribute Sets, whose Attribute Values are logically related and cannot be separated. They have a meaning only as one entity and so, if separated, their semantics will be lost. Non-Decomposable Nested attribute(s) can, but need not necessarily, be part of the key of the nested relation.

MODULE′	**COURSE**
Pascal	Programming
C++	Programming
Word 97	Word processor
Word Perfect	Word processor
Microsoft Works	Word processor
Access	Database and Spreadsheet
Excel	Database and Spreadsheet
Java	Programming
Visual Basic	Programming
Pascal	Programming

Figure 3. UNNEST($_{\text{MODULE(MODULE')}}$SCHOOL)

Definition 10 (*Non-Decomposable Nested attribute*) Let R be a nested relation. Assume A is a nested attribute of R consisting of attributes $A_1, A_2, ..., A_n$ ($n \geq 1$) and B an atomic attribute of R. Let also, A functionally determines attribute B (i.e., $A \rightarrow B$). A is a Non-Decomposable Nested attribute of R iff UNNEST($_{A(A_1, A_2, ..., A_n)}$R), then $A_1, A_2, ..., A_n \rightarrow B$ does not hold (i.e., $A_1, A_2, ..., A_n \not\rightarrow B$).

Example 3.
The EXAMS relation in Figure 4 consists of two attributes: the Non-Decomposable Nested attribute SUBJECT and the atomic attribute DEGREE. The semantics of this relation is that a number of specific subjects are needed for a degree. The Non-Decomposable Nested attribute SUBJECT is the key attribute of the relation and therefore, SUBJECT $\rightarrow$ DEGREE. The result of attempting to unnest the EXAMS relation is shown in Figure 5. However, the result relation is not a valid relation because we lose information about that functional dependency. Specifically, if UNNEST($_{\text{SUBJECT(SUBJECT')}}$EXAMS), then SUBJECT' $\not\rightarrow$ DEGREE.

SUBJECT	DEGREE
SUBJECT′	
Composition	
Maths	Science
Physics	
Composition	
Statistics	Economics
Maths	
Composition	
Maths	Science
Chemistry	
Composition	
History	Arts
Latin	
Ancient Greek	

Figure 4. The nested relation EXAMS

SUBJECT′	DEGREE
Composition	Science
Maths	Science
Physics	Science
Composition	Economics
Statistics	Economics
Maths	Economics
Composition	Science
Maths	Science
Chemistry	Science
Composition	Classics
History	Classics
Latin	Classics
Ancient Greek	Classics

Figure 5. UNNEST($_{\text{SUBJECT(SUJBECT')}}$EXAMS)

The above definitions express necessary and sufficient conditions for the discrimination between Decomposable and Non-Decomposable attributes.

3 UNNESTING AND NESTING IN NESTED RELATIONS

Fischer and Thomas (1983), Jaeschke and Schek (1982), Schek and Scholl (1986), and Thomas and Fischer (1986) mention that unnesting a nested relation R and then nesting it on the same attribute does not always give the original relation R. In Fischer and Thomas (1983) the following example (Figure 6) is given as a counterexample to show that the equality $\text{NEST}_{B=(B')}(\text{UNNEST}_{B}(R)) = R$ does not necessarily hold at all times. Jaeschke and Schek (1982) prove that this equality does not hold when a nested relation is not "nested completely" along the nested attribute. In other words, when a nested attribute is also a key attribute, the nest operation is not the inverse of the unnest operation. In Figure 6, relation R is not "nested completely" along attribute B because the two nested tuples of relation R, having the same data value in the atomic attribute A, should form one nested tuple. To overcome this problem, several researchers (Abiteboul & Bidoit, 1983; Deshpande & Larson, 1991; Roth, Korth & Silberschatz, 1988) have suggested that nested relations should be in Partitioned Normal Form (PNF) that means that all or a subset of the flat attributes of the relation should form a key for the relation, and recursively, each nested attribute of a relation is also in Partitioned Normal Form. However, this is an undesirable restriction that is difficult to apply universally because occasionally it might be preferable to have nested attributes as key attributes in a relation (Garani, 2003).

A	B
	B′
0	0
	1
0	1
	2

Figure 6. Structure where NEST is not inverse to the UNNEST (Fischer & Thomas, 1983)

In Section 2.1 of this paper, it has been shown that unnesting a Non-Decomposable Nested attribute is invalid semantically. This is because unnesting a Non-Decomposable Nested attribute does not preserve the functional dependency between that attribute and the remaining ones. Thus, information of relatedness property between data items of the relation is lost. On the contrary, if the nested attribute that has to be unnested is a Decomposable Nested attribute, the functional dependency is maintained. In this case, it should also be noted that two Decomposable Nested Attribute Elements belonging to different Decomposable Nested Attribute Sets do not have to be different, but if such a relation is unnested and then nested on the same attribute, information is not lost, although duplicate Decomposable Nested Attribute Elements will be eliminated.

Theorem 1. Let R be a nested relation and A a Decomposable Nested Attribute of R consisting of attributes A_1, A_2, …, A_n. Then,

$$\text{NEST}_{A=(A_1, A_2, \ldots, A_n)}(\text{UNNEST}_{A(A_1, A_2, \ldots, A_n)}(R))=R$$

Proof. According to Definition 7, because A is a Decomposable Nested attribute of R and A functionally determines attribute B (i.e. $A \rightarrow B$), $\text{UNNEST}(_{A(A_1, A_2, \ldots, A_n)}R)= R'$, and $A_1, A_2, \ldots, A_n \rightarrow B$.
Therefore, since $A_1, A_2, \ldots, A_n \rightarrow B$ then, $\text{NEST}_{A=(A1, A2, \ldots, An)}(R')=R$.

In Figure 6, because the semantics of the data in the relation are not known, two cases must be distinguished: 1) the nested attribute B is a Decomposable Nested attribute and 2) the nested attribute B is a Non-Decomposable Nested attribute. In the first case, the key attribute B consists of sets of data values that form Decomposable Nested Attribute Sets. Even when the same Decomposable Nested Attribute Elements are in two or more different Decomposable Nested Attribute Sets, as in the relation of Figure 6, unnesting and then nesting can be performed without information loss, because the same data values of key attribute B have the same attribute value for A in the different tuples. Therefore, in this case, the result of an unnest operation followed by a nest operation does not change the information held in the original relation. However, as mentioned earlier, duplicate data may be eliminated.

In the second case, where B is a Non-Decomposable Nested attribute, unnesting should not be allowed at all. Therefore, "nested completely relations" does not mean anything more than that Decomposable Nested Attribute Sets are combined (merged) such that duplicate values are eliminated. In fact, this is only a representation issue as there is no change of the information content of the relation.

Deshpande & Larson (1991) discuss the problem of having nested attributes as part of the key when an unnest operation is taking place, since information about the functional dependency is lost. In fact, this applies only in the case of Non-Decomposable Nested attributes for which it has been demonstrated above that the semantics of the relation are totally lost, so unnesting should not be performed at all.

Makinouchi allows the left hand side of a functional dependency to be a relation. The following example is given by Makinouchi (1977) where this type of dependency is shown.

Example 4. Relation U_TRIANGLE(3_POINTS(X, Y), AREA, COLOR) (Figure 7) records the size and color of triangles. The dependency 3_POINTS → AREA, COLOR means that one triangle has one area and one color.

3_POINTS		AREA	COLOR
X	Y		
1	1	2	red
3	1		
3	3		

Figure 7. The nested relation U_TRIANGLE

In Ling & Yan (1994), the above example is discussed. Two problems are considered. The first one concerns the fact that in this particular example, nested relation 3_POINTS should contain exactly three tuples. The second one is that according to Ling & Yan (1994), the above dependency is difficult to justify, and therefore, the attribute 3_POINTS is better considered as an atomic attribute with a type such as "3 ordered integer pairs" rather than a nested attribute.

However, we can provide as a counterexample relation EXAMS (Figure 4) where nested attribute SUBJECT contains different number of tuples, and therefore, the number of tuples should not be fixed. Furthermore, according to our definition, attribute 3_POINTS is a Non-Decomposable Nested attribute, and relation U_TRIANGLE becomes nonsense after being flattened because the functional dependency X, Y $\rightarrow$ AREA, COLOR does not hold (X, Y $\nrightarrow$ AREA, COLOR), in a way analogous to attribute SUBJECT in EXAMS relation (SUBJECT' $\nrightarrow$ DEGREE).

In summary, the proposition that unnesting and then nesting a nested relation R on the same attribute does not, necessarily, give the original relation R has been shown to be incorrect when the semantic properties of the data are taken into account. In particular, cases such as the "counterexample" in Figure 6 either cannot be unnested, or if it can be, then no information is lost when the unnest operation and the subsequent nest operation are performed. Thus, for all semantically meaningful relations, unnesting and then nesting on the same attribute will yield the original relation subject only to the elimination of duplicate data.

4 CONCLUSION

This paper has shown that nested attributes can be partitioned into two categories, Decomposable Nested attributes and Non-Decomposable Nested attributes, according to their semantics. This classification has been used in this paper to demonstrate that for all nested relations, the effect of an unnest operation can always be undone by a subsequent nest operation on the same attribute, without loss of information and with, at most, the elimination of duplicate tuples. Therefore, the two operations are inverse, and, as a consequence, there is always a sequence of nest operations that will be an inverse for any sequence of valid unnest operations. The previous sentence means that query optimisation is not impeded. Furthermore, the nested relational algebra can be defined by unnesting the involved relations into the 1NF, applying some basic algebraic operations, and finally, converting the result back to nested relations. In other words, the operators of the nested relational algebra behave in the same way as it would be in 1NF.

5 REFERENCES

Abiteboul, S., & Bidoit, N. (1983) Non First Normal Form Relations: An Algebra Allowing Data Restructuring. *Journal of Computer and System Sciences 33*, 361-393.

Deshpande, V., & Larson, P.A. (1991) An Algebra for Nested Relations with Support for Nulls and Aggregates. Technical Report CS-91-16, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.

Fischer, P.C., & Thomas, S.J. (1983). Operators for Non-First-Normal Form Relations. *Proc. of the IEEE Computer Society's 7th International Conference on Computer Software and Applications* (pp. 464-475). Chicago, Illinois, USA.

Garani, G. (2006) A Generalised Relational Data Model Approach in the Organisation and Management of Nested Data. *Journal Annals of Mathematics, Computing & Teleinformatics 1*(4), 17-23.

Garani, G. (2003) *A Temporal Database Model Using Nested Relations*, PhD thesis, Dept. of Computer Science, Birkbeck College, Univ. of London, U.K.

Gyssens, M., Suciu, D., & Van Gucht, D. (2001) Equivalence and Normal Forms for the Restricted and Bounded Fixpoint in the Nested Algebra. *Information and Computation 164*(1), 85-117.

Hernández, H.J., (1993) Extended Nested Relations. *Acta Informatica 30*, 741-771.

Jaeschke, G., & Schek, H.J. (1982) Remarks on the Algebra of Non First Normal Form Relations. *Proc. of the ACM Symposium on Principles of Database Systems* (pp. 124-138). Los Angeles, California, USA.

Ling, T.W., & Yan, L.L. (1994) NF-NR: A Practical Normal Form for Nested Relations. *Journal of Systems Integration 4*, 309-340.

Liu, J., Vincent, M., & Mohania, M. (1999) Incremental evaluation of nest and unnest operators in nested relations. *Proc. of the Second Intl. Symposium on Cooperative Database Systems for Advanced Applications* (pp. 264-275). Wollongong, Australia.

Makinouchi, A. (1977) A consideration on Normal Form of Not-Necessarily-Normalized Relations in the Relational Data Model. *Proc. of the International Conference on VLDB* (pp. 447-453). Tokyo, Japan.

Roth, M.A., Korth H.F., & Silberschatz, A. (1988) Extended Algebra and Calculus for Nested Relational Databases. *ACM Transactions on Database Systems 13*, 389-417.

Schek, H.-J., & Scholl, M.H. (1986) The Relational Model with Relation-Valued Attributes. *Information Systems 11*, 137-147.

Thalheim, B. (2000) *Entity-Relationship Modeling. Foundations of Database Technology*. Springer-Verlag.

Thomas, S.J., & Fischer, P.C. (1986) Nested Relational Structures. *Advances in Computing Research 3*, 269-307.

Ullman, J.D. (1988) *Principles of Database and Knowledge-Base Systems*, Rockville, Maryland: Computer Science Press.