Abstract

Unnormalized relations permit components of tuples to be relation instances themselves instead of atomic values. Such structures do not always represent the restructuring of a flat (1NF) relation. It is shown that for one-level structures (nesting is permitted only over sets of attributes) there is a polynomial time algorithm to determine whether the structure is the result of restructuring a flat relation with a sequence of NEST operations.

1. Introduction

Most work on the relational model of Codd [Cod] has involved the first normal form (1NF) assumption, i.e., that all elements of a tuple of a relation are atomic values. However, difficulties of modelling the real world using 1NF have led to investigations of ways of relaxing 1NF while retaining much of the advantages of the relational model. Generalizations have proceeded in two directions: allowing null values and allowing richer structures as entries in a tuple, e.g., a data item might itself be a relation.

Early work in the second direction by Makinouchi has led to the concept of nesting [Mak]. This was later studied by Jaeschke and Schek for one-level nesting [JS] over a single attribute and by Thomas and Fischer [TF,Tho] in a more general setting. Kambayashi, Tanaka and Takeda considered nesting (called *row-nest* in [KTT]) as well as other related operators for restructuring relations without loss of information. Abiteboul and Bidoit considered multi-level nesting under very strong conditions so that the unnested attributes of a relation would be a key for the relation [AB]. They also permitted null values. Roth, Korth and Silberschatz have used similar assumptions in their work but do not permit null values [RKS].

We wish to study non-1NF relations under as few assumptions as possible. For example, the assumptions in [AB] and [RKS] are sufficient, but not necessary, to guarantee that all NEST operations on the same level commute. A complete characterization of the permutability of one-level nesting is given by the authors in [FV1] and an efficient algorithm for detecting this condition is given in [FV2]. In this paper we will not assume that two different NEST operations necessarily commute.

We present an informal overview of our subject before giving the formal development. A structure (called *nested relational structure* in [TF]) is simply an unnormalized relation scheme and instance, i.e., a relation in the original sense of [Cod]. In a structure, elements of a tuple may either be atomic or may be relations, which themselves need not be in 1NF.
Example 1: An example of a structure is shown in Figure 1.

<table>
<thead>
<tr>
<th>PARENT</th>
<th>CHILD</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>e1</td>
</tr>
<tr>
<td></td>
<td>(h1, h2)</td>
</tr>
<tr>
<td></td>
<td>(h3)</td>
</tr>
<tr>
<td>f2</td>
<td>e3</td>
</tr>
<tr>
<td></td>
<td>(h1, h3)</td>
</tr>
<tr>
<td></td>
<td>{hi}</td>
</tr>
<tr>
<td></td>
<td>(h4)</td>
</tr>
</tbody>
</table>

We shall use the term relation or flat relation to mean a 1NF relation, i.e., all data items in tuples are atomic. A relation is, of course, a special case of a structure. We allow NEST (v) and UNNEST (µ) operations on any structure; formal definitions are presented below. Essentially, nesting on a set of objects Y collects into sets those subtuples over Y which agree on all values outside of the nesting attributes Y. Unnesting will reverse this process.

Example 2: Consider the flat relation r2 shown in Figure 2. A tuple (p,c) belongs to r2 if p is a natural parent of child c. In Figures 3 through 6 below we show r3 = νPARENT(r2), r4 = νCHILD(r2), r5 = νCHILD(νPARENT(r2)), and r6 = νPARENT(νCHILD(r2)) respectively.

<table>
<thead>
<tr>
<th>PARENT</th>
<th>CHILD</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
</tr>
<tr>
<td>p3</td>
<td>c3</td>
</tr>
<tr>
<td>p4</td>
<td>c4</td>
</tr>
<tr>
<td>p5</td>
<td>c4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PARENT*</th>
<th>CHILD*</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>{c1, c2}</td>
</tr>
<tr>
<td>p2</td>
<td>{c1, c2}</td>
</tr>
<tr>
<td>p3</td>
<td>{c3}</td>
</tr>
<tr>
<td>p4, p5</td>
<td>{c4}</td>
</tr>
</tbody>
</table>

Figure 1

This example illustrates that nesting does not necessarily commute. Furthermore, notice how subtle semantic relationships existing between parents and children can be highlighted when we use structures with set-valued entries. This suggests that the 1NF assumption may be too strong for database design.

Between relations and arbitrary structures there are many classes of structures; we discuss two here. A normalization-lossless (NL) structure is a structure that can be fully unnested to a relation and recreated from the flat relation using only legal NEST and UNNEST operations. This was the main class studied in [TF, Tho]. Intuitively, by normalizing such a structure (i.e., bringing it to 1NF), no information is lost. Clearly, the class of all structures and the class of NL structures are each closed under NEST and UNNEST operations.

Example 3: The structure r7 = νPARENT*(r5) shown in Figure 7 is an example of a NL structure. Figures 2 through 6 also depict NL structures.

Example 4: An example of a structure r8 which is not a NL structure is shown in Figure 8 (cf. [Mak]). A tuple (T,A) belongs to r8 if A is the area of the triangle T.
The reader is invited to verify that \( \text{point}(\text{point}^*(r_8)) \) is the structure \( r_9 \) shown in Figure 9. This example illustrates that information can be lost through normalization, a situation which clearly should be avoided.

A nested relation is a structure which can be obtained from a relation using only legal NEST operations. All structures considered in Example 2 are examples of nested relations. The structures used in [AB] and the *partitioned-normal-form relations* discussed in [RKS] are a proper subclass of the class of nested relations. Nested relations can often provide a succinct representation of flat relations and can be a useful alternative to vertical decomposition in dealing with update and other anomalies (cf. [KTT, FV1, FV2, FV3]).

While the class of nested relations is obviously closed under NEST operations, it is easy to show they are not closed under UNNEST. First we observe that the structure \( r_7 \) in Figure 7 above is a NL structure which is not a nested relation. Since unnesting \( r_7 \) produces the relation \( r_2 \) in Figure 2, we see that if \( r_7 \) were a nested relation it would be equal to \( r_4 \), which is not the case. From the definition of \( r_7 \) in Example 3 we see that an UNNEST operation on \( r_5 \), a nested relation, produces \( r_7 \), which is not a nested relation.

The question naturally arises as to when a given structure is in fact a nested relation. This question is obviously recursively solvable in exponential time since one need only unnest the structure to a flat relation and try all possible legal renesting combinations, comparing each result with the original structure. Thus, we are interested in whether a polynomial-time algorithm exists. We are able to do this for the special case of one-level nested relations, i.e., structures where nesting is done only over sets of basic attributes. In this class, elements of a tuple are either atomic or may be 1NF relations.

In our development we will introduce a generalization of the notion of functional dependency which appropriately deals with the effects of nesting. We will show that the order of nesting creates a *trace* of these strong functional dependencies. From this trace one can gain information about sequences of nest operations which could recreate the given structure.

2. Basic Concepts

In order to formalize the notion of structures, we need the auxiliary concept of scheme. Intuitively, a scheme specifies the underlying frame of a structure.

Let \( U \) denote the universe of attributes.

A scheme \( R \) and its set of underlying attributes, \( \text{attr}(R) \), is defined recursively by:

1. If \( R \) is a finite subset of \( U \) then \( R \) is a (flat) scheme and \( \text{attr}(R) = R \).
2. Let \( X \) be a flat scheme and \( Y_1, \ldots, Y_n \) schemes such that \( \text{attr}(X), \text{attr}(Y_1), \ldots, \text{attr}(Y_n) \) are pairwise disjoint, then \( R = X \cup \{Y_1, \ldots, Y_n\} \) is a scheme, and \( \text{attr}(R) = X \cup \bigcup_{i=1}^{n} \text{attr}(Y_i) \).

Let \( R \) be a scheme, then \( A(R) = R \cap U \) is called the set of (basic) attributes of \( R \) and \( H(R) = R - A(R) \) is called the set of higher order objects of \( R \).

As an example, let \( U = ABCDEFG \). (Note that we are adopting the well-known but imprecise convention of sometimes representing sets by concatenating their members and omitting outside braces and also using concatenation to represent set union.) Then for the scheme \( R = \{A, B, \{C, D\}, \{E, \{F\}\}\} \), we have \( A(R) = AB \) and \( H(R) = \{\{C, D\}, \{E, \{F\}\}\} \).
The reader may note the ambiguity in whether the set \{C, D\} refers to the set CD of two basic attributes or to a single higher order object. In [TF] and [RKS] this ambiguity is resolved by adding *rules* to the formal system which explicitly name the higher order objects. In [AB] and [KTT] this ambiguity is resolved by treating a scheme as a string rather than a set of objects. We choose to resolve this ambiguity, when necessary, by tagging a set or a variable with a superscript '*' to indicate that it represents a higher order object. Thus, if we write \{C, D\}* we refer to a member of H(R), but if we write \{C, D\} or CD we refer to a set of two attributes. When context makes the meaning clear, we omit the '*', e.g., let M be a member of H(R)*.

We are now able to define structures (instances) over a scheme R. We assume each attribute A \in U has a set of values associated with it, called the domain of A and denoted dom(A).

Let R be a scheme. The set of structures over R, denoted str(R), is recursively defined by:

1. If R is a flat scheme, then r is in str(R) if r is a finite nonempty set of tuples over R, where a tuple t over R is a mapping from R into \bigcup_{A \in R} dom(A) such that t(A) \in dom(A) for each A \in R.

2. If R is a scheme, then r is in str(R) if r is a finite nonempty set of tuples over R, where a tuple t over R is a mapping from R into \left(\bigcup_{A \in R} dom(A) \cup \bigcup_{M \in W} str(M)\right) such that t(A) \in dom(A) for each A \in A(R) and t(M) \in str(M) for M \in H(R).

We now define the NEST and UNNEST operators. First, let t be a tuple over scheme R and X \subseteq R. The X-value of t, denoted t[X], is the restriction of the mapping t to X. Note that some members of X may be higher order objects.

Let r be a structure over scheme R and let Y \subseteq R. Then \nu_Y(r) is a structure over (R \setminus Y)Y* such that a tuple v \in \nu_Y(r) if and only if:

1. there exists a tuple t \in r such that t[R \setminus Y] = v[R \setminus Y] and
2. v(Y*) = \{ t'[Y] \mid t' \subseteq r \text{ and } t'[R \setminus Y] = v[R \setminus Y] \}.

Let s be a structure over scheme S and let Y* \subseteq H(S). Then \mu_{Y*}(s) is a structure over (S \setminus Y*)Y such that t \in \mu_{Y*}(s) if and only if there exists a tuple v \in s such that t[S \setminus Y'] = v[S \setminus Y'] and t[Y] \in v(Y*).

3. Strong Functional Dependencies

Let r be a structure over scheme R. Let V \subseteq R, W \subseteq H(R), and Z \subseteq R. Furthermore let V \cap W = \emptyset. Then r satisfies the strong functional dependency (SFD) V\prec W \rightarrow\succ Z if and only for any two tuples t_1, t_2 \in r such that t_1[V] = t_2[V] and t_1[M] \cap t_2[M] \neq \emptyset for each M \in W, we have t_1[Z] = t_2[Z].

When W = \emptyset, a strong functional dependency is nothing but an ordinary functional dependency (FD) and we shall use standard FD notation [Ull].

Example 5: The structure r_{10} shown in Figure 10 does not satisfy the SFD A\prec B* \rightarrow\succ C.

\[
\begin{array}{ccc}
A & B^* & C \\
\hline
a & \{b_1, b_2\} & c_1 \\
\{b_2, b_3\} & c_2 \\
\end{array}
\]

Figure 10

Notice however that r_{10} satisfies the SFD AB* \rightarrow\succ C.

Remark 1: Let r be a structure over scheme R. Let V, Z \subseteq R, and Y* \subseteq W \subseteq H(R). If r satisfies the SFD V\prec W \rightarrow\succ Z then r satisfies the SFD VY*\prec (W-Y*) \rightarrow\succ Z.

The proof is immediate from the definition of a SFD. The converse does not hold, cf. Example 5.

174
Our theory uses four basic results concerning \( \nu, \mu \) and SFDs. The first three were proved in [TF].

**Lemma 1:** Let \( r \) be any structure over scheme \( R \) and let \( Y \subseteq R \). Then
\[
\mu_Y(\nu_Y(r)) = r
\]

**Lemma 2:** Let \( s \) be a structure over scheme \( S \) and let \( M, N \in H(S) \). Then
\[
\mu_M(\mu_N(s)) = \mu_N(\mu_M(s)).
\]

**Remark 2:** In view of Lemma 2, the result of unnesting on more than one higher object is independent of the order of unnesting. We therefore extend our notation so that
\[
\mu_{M_1 M_2 \ldots M_k}(s) = \mu_{M_k}(\mu_{M_{k-1}}(\ldots(\mu_{M_1}(s))\ldots))
\]
where \( s \) is a structure over \( S \) and \( M_1 M_2 \ldots M_k \in H(S) \).

**Lemma 3:**

a. Let \( r \) be a structure over scheme \( R \) and let \( Y \subseteq R \). Then \( (R - Y) \rightarrow Y^* \) holds in \( \nu_Y(r) \).

b. Let \( s \) be a structure over scheme \( S \) and let \( Y^* \in H(S) \). Then \( (S - Y^*) \rightarrow Y^* \) holds in \( s \) if and only if \( \nu_Y(\mu_Y(s)) = s \).

**Lemma 4:**

a. Let \( r \) be a structure over scheme \( R \). Let \( X, Y, Z \subseteq R \) and let \( W \subseteq H(R) \). Then \( r \) satisfies \( XY < W > \rightarrow Z \) if and only if \( \nu_Y(\mu_Y(\mu_Y(\mu_Y(\ldots(s))\ldots))) \) exists.

b. Let \( s \) be a structure over scheme \( S \). Let \( X, Y \subseteq S \), let \( Y^* \in H(S) \) and \( W \subseteq (H(S) - Y^*) \). Then \( s \) satisfies \( XY < W > \rightarrow Z \) if and only if \( \nu_Y(\mu_Y(s)) = s \).

**Proof:**

a. If \( r \) does not satisfy \( XY < W > \rightarrow Z \) then there exist \( t_1, t_2 \in r \) such that \( t_1[X] = t_2[X] \), \( t_1[Y] \neq t_2[Y] \), \( t_1[M] \neq t_2[M] \) for each \( M \in W \), and \( t_1[Z] \neq t_2[Z] \). Hence in \( \nu_Y(r) \) there exist two distinct tuples \( v_1, v_2 \) such that \( v_1[X] = v_2[X], v_1[Y] \neq v_2[Y], v_1[M] \neq v_2[M] \) if and \( v_1[Z] \neq v_2[Z] \) which violates \( XY < W > \rightarrow Z \) in \( \nu_Y(r) \).

If \( \nu_Y(r) \) does not satisfy \( XY < W > \rightarrow Z \) then there exist \( v_1, v_2 \in \nu_Y(r) \) such that \( v_1[X] = v_2[X], v_1[Y] \neq v_2[Y], v_1[M] \neq v_2[M] \) for each \( M \in W \), and \( v_1[Z] \neq v_2[Z] \). By Lemma 1, \( \nu_Y(\nu_Y(r)) = r \) and we therefore find \( t_1, t_2 \in r \) such that \( t_1[X] = t_2[X], t_1[Y] = t_2[Y], t_1[M] \neq t_2[M] \) for each \( M \in W \), and \( t_1[Z] \neq t_2[Z] \), which violates \( XY < W > \rightarrow Z \).

b. If \( s \) does not satisfy \( XY < W > \rightarrow Z \) then there exist \( v_1, v_2 \in s \) such that \( v_1[X] = v_2[X], v_1[Y] \neq v_2[Y], v_1[M] \neq v_2[M] \) for each \( M \in W \), and \( v_1[Z] \neq v_2[Z] \). Clearly in \( \mu_Y(s) \) there will be two tuples \( t_1, t_2 \) such that \( t_1[X] = t_2[X], t_1[Y] = t_2[Y], t_1[M] \neq t_2[M] \) if and \( t_1[Z] \neq t_2[Z] \), which violates \( XY < W > \rightarrow Z \) in \( \nu_Y(s) \).

If \( \mu_Y(s) \) does not satisfy \( XY < W > \rightarrow Z \) then there exist \( v_1, v_2 \in \mu_Y(s) \) such that \( v_1[X] = v_2[X], t_1[Y] = t_2[Y], t_1[M] \neq t_2[M] \) for each \( M \in W \), but \( t_1[Z] \neq t_2[Z] \). Since \( t_1[Z] \neq t_2[Z] \) these tuples must have come from two different tuples \( v_1, v_2 \) in \( s \). Furthermore, these tuples satisfy \( v_1[X] = v_2[X], t_1[Y] \neq v_2[Y], t_1[M] \neq v_2[M] \) if and \( t_1[Z] \neq v_2[Z] \), which violates \( XY < W > \rightarrow Z \) in \( s \).

**4. One-Level Nested Relations**

Let \( S = X_1^* \ldots X_n^* \) be a scheme, i.e., \( X = A(S) \) and \( Y_1^* \ldots Y_n^* = H(S) \). We say that \( S \) is a one-level scheme if \( Y_i \) is a flat scheme for each \( i, 1 \leq i \leq n \). Let \( s \) be a structure over a one-level scheme \( S \). We say \( s \) is a one-level nested relation (1NR) if and only if there exists a relation \( r \) over \( \text{attr}(S) \) and a...
permutation $(i_1,i_2,...,i_n)$ of the integers $1,2,...,n$ such that
\[ s = \nu_n^{i_1} (\nu_n^{i_2} (\ldots (\nu_n^{i_n} (r) \ldots ))) \]

Remark 3: It follows from Lemma 1 and Remark 2 that $r = \mu_{H(S)}(s)$.

The following theorem gives a characterization of a one-level nested relation in terms of the SFDs holding in the structure.

Theorem 1: Let $s$ be a structure over a one-level scheme $S = \{X_1,...,X_n\}$. Let
\[ s_i = \mu_{Y_{i-n+1}} (s) \]
for $1 \leq i \leq n$. Thus, $s_n$ is the normalized relation obtained by completely unnesting $s$, i.e.,
\[ s_n = \mu_{H(S)}(s) \].
Then
\[ s = \nu_n (\nu_n^{i_1} (\ldots (\nu_n^{i_n} (s) \ldots ))) \]
if and only if
\[ s \text{ satisfies } XY_1^*...Y_{i-1}^* < Y_{i+1}^*...Y_n^* \rightarrow Y_j^* \]
for each $j$, $n-i+1 \leq j \leq n$.

Proof: The proof will be by induction on $i$.

Basis: $i = 1$. Then from Lemma 3b
\[ s \text{ satisfies } XY_1^*...Y_{i-1}^* \rightarrow Y_i^* \text{ if and only if } s = \nu_n (\nu_n^{i_1} (s)) = \nu_n (s_1). \]

Induction Step: Assume that for all $m$, $1 \leq m < n$, the theorem holds, i.e.,
\[ s = \nu_n (\nu_n^{i_1} (\ldots (\nu_n^{i_m} (s_m) \ldots ))) \]
if and only if
\[ s \text{ satisfies } Y_1^*...Y_{i-1}^* < Y_{i+1}^*...Y_n^* \rightarrow Y_j^* \]
for each $j$, $n-m+1 \leq j \leq n$.

Let $i = m + 1$. We have to prove that:
\[ s = \nu_n (\nu_n^{i_1} (\ldots (\nu_n^{i_m} (\nu_n^{i_{m+1}} (s_m)) \ldots ))) \]
if and only if $s$ satisfies the SFDs
\[ XY_1^*...Y_{i-1}^* < Y_{i+1}^*...Y_n^* \rightarrow Y_j^* \]
for $n-m \leq j \leq n$.

We have:
\[ s = \nu_n (\nu_n^{i_1} (\ldots (\nu_n^{i_m} (\nu_n^{i_{m+1}} (s_m)) \ldots ))) \]
if and only if
\[ s_m = \nu_n (\nu_n^{i_{m+1}} (s_m)) \]
and
\[ s = \nu_n (\nu_n^{i_1} (\ldots (\nu_n^{i_m} (s_m) \ldots ))) \].
This holds if and only if $s_m$ satisfies the SFD
\[ XY_1^*...Y_{n-m+1}^* Y_{n-m+1}^*...Y_n^* \rightarrow Y_{n-m}^* \]
by Lemma 3b, and $s$ satisfies the SFDs
\[ XY_1^*...Y_{j-1}^* < Y_{j+1}^*...Y_n^* \rightarrow Y_j^* \]
for each $j$, $n-m+1 \leq j \leq n$, by the induction hypothesis. To complete the proof we need only observe by repeated application of Lemma 4b that (1) holds if and only if $s$ satisfies the SFD
\[ XY_1^*...Y_{n-m-1}^* < Y_{n-m+1}^*...Y_n^* \rightarrow Y_{n-m}^* \]

Theorem 2: Let $s$ be a structure over a one-level scheme $S = \{X_1,...,X_n\}$. The following statements are equivalent:

1. For any permutation $(i_1,i_2,...,i_n)$ of the integers $1,2,...,n$
\[ s = \nu_n (\nu_n^{i_1} (\ldots (\nu_n^{i_n} (s) \ldots ))) \]
if and only if
\[ s \text{ satisfies } XY_1^*...Y_{i-1}^* < Y_{i+1}^*...Y_n^* \rightarrow Y_j^* \]
for each $j$, $1 \leq j \leq n$.

2. The structure $s$ satisfies the SFDs
\[ X<Y_1^*...Y_{j-1}^* Y_{j+1}^*...Y_n^* \rightarrow Y_j^* \]
for each $j$, $1 \leq j \leq n$.

3. For any two tuples $v_1, v_2 \in s$ with
\[ v_1[X] = v_2[X] \]
there exist indices $p$ and $q$, $1 \leq p < q \leq n$, such that
\[ v_1[Y_p^*] \cap v_2[Y_p^*] = \emptyset \text{ and } v_1[Y_q^*] \cap v_2[Y_q^*] = \emptyset. \]

Proof: 1 $\Rightarrow$ 2. Since $s = \nu_n (\ldots \nu_n^{i_1} (\nu_n^{i_{j-1}} (\nu_n^{i_j} (\nu_n^{i_{j+1}} (\ldots (\nu_n^{i_n} (s) \ldots )\ldots ))))$ for each $j$, $1 \leq j \leq n$, we know from Theorem 1 that $s$ satisfies the SFD
\[ X<Y_1^*...Y_{j-1}^* Y_{j+1}^*...Y_n^* \rightarrow Y_j^* \]
for each $j$, $1 \leq j \leq n$.

2 $\Rightarrow$ 1. Consider any permutation
of the integers 1, 2, ..., n. Since $s$ satisfies the SFD

$$X < Y_1^* \ldots Y_j^* Y_{j+1}^* \ldots Y_n^* > \rightarrow Y_j^*$$

for each $j$, $1 < j < n$, by Remark 1, $s$ also satisfies the SFDs

$$XY_{i_1}^* \ldots Y_{i_j}^* < Y_{i_{j+1}}^* \ldots Y_{i_n}^* > \rightarrow Y_{i_j}^*$$

for each $j$, $1 \leq j \leq n$. By Theorem 1 we obtain

$$s = \nu_{Y_{i_n}}(\nu_{Y_{i_{n-1}}}(...(\nu_{Y_{i_2}}(\nu_{Y_{i_1}}(\mu_{H(S)}(s))))...))$$

2 $\iff$ 3. Follows immediately from the definition of the SFDs.

**5. A Polynomial-Time Algorithm for Identifying Nested Relations**

In this section, we will develop an algorithm to test whether a structure over a one-level nested scheme is a one-level nested relation. We first prove two technical lemmas.

**Lemma 5:** Let $s$ be a structure over a one-level scheme $XZ_1^* \ldots Z_n^*$. Let $s_i$ be as in Theorem 1, for $2 \leq i \leq u$ and let $k = n-i+1$. If $s$ satisfies the SFDs

$$XY_{i_1}^* \ldots Y_{i_k}^* < Y_{i_{k+1}}^* \ldots Y_{i_n}^* > \rightarrow Y_{i_k}^*$$

and

$$XY_{i_1}^* \ldots Y_{i_k}^* < Y_{i_{k+1}}^* \ldots Y_{i_n}^* > \rightarrow Y_{i_{k+1}}^*$$

then $\nu_{Y_{i_{k+1}}}((\nu_{Y_{i_k}}(s_i))) = \nu_{Y_{i_k}}((\nu_{Y_{i_{k+1}}}(s_i)))$.

**Proof:** By Lemma 4b, $s_{i-2} = \nu_{Y_{i_2}}(\nu_{Y_{i_1}}(s))$ satisfies the SFDs

$$XY_{i_1}^* \ldots Y_{i_{k+1}}^* < Y_{i_{k+2}}^* \ldots Y_{i_n}^* > \rightarrow Y_{i_{k+1}}^*$$

and

$$XY_{i_1}^* \ldots Y_{i_{k+1}}^* < Y_{i_{k+2}}^* \ldots Y_{i_n}^* > \rightarrow Y_{i_{k+2}}^*$$

By Remark 1, $s_{i-2}$ also satisfies the SFDs

$$XY_{i_1}^* Y_{i_k}^* Y_{i_{k+1}}^* Y_{i_{k+2}}^* \ldots Y_{i_n}^* \rightarrow Y_{i_k}^*$$

and

$$XY_{i_1}^* Y_{i_{k+1}}^* Y_{i_{k+2}}^* \ldots Y_{i_n}^* \rightarrow Y_{i_{k+1}}^*$$

Recall that $\nu_{Y_{i_{k+1}}}((\nu_{Y_{i_k}}(s_i))) = s_i$. By Theorem 1, $s_{i-2} = \nu_{Y_{i_{k+1}}}((\nu_{Y_{i_k}}(s_i)))$ since $s_{i-2}$ satisfies the SFDs (2) and (5) and $s_{i-2} = \nu_{Y_{i_{k+1}}}((\nu_{Y_{i_k}}(s_i)))$ since $s_{i-2}$ satisfies the SFDs (3) and (4). Hence we obtain the desired equality.

Let $s$ be a structure over one-level scheme $XZ_1^* \ldots Z_n^*$. Let $P = (i_1, i_2, \ldots, i_u)$ be a permutation of the integers $1, 2, \ldots, n$. $(Y_{i_1}, Y_{i_2}, \ldots, Y_{i_u})$ is called a **nesting sequence** for $r$ if $s = \nu_{Y_{i_u}}(\nu_{Y_{i_{u-1}}}(\nu_{Y_{i_{u-2}}}(\nu_{Y_{i_{u-3}}}(\mu_{H(S)}(s))))))$. By Theorem 1 we know that $s$ satisfies the SFD

$$X < Z_1^* \ldots Z_{k-1}^* Z_k^* Z_{k+2}^* \ldots Z_n^* > \rightarrow Z_k^*$$

By our hypothesis $s$ satisfies the SFD

$$X < Z_1^* \ldots Z_{k-1}^* Z_k^* Z_{k+2}^* \ldots Z_n^* > \rightarrow Z_{k+1}^*$$

hence by Remark 1, $s$ satisfies the SFD

$$X < Z_1^* \ldots Z_{k-1}^* Z_k^* Z_{k+2}^* \ldots Z_n^* > \rightarrow Z_{k+1}^*$$

Let $i = n-k+1$. Consider

$$s_i = \nu_{Z_{n-1}^* \ldots Z_{k+2}^* \ldots Z_n^*}(s).$$

Clearly,

$$s_i = \nu_{Z_{k-1}^*}(\nu_{Z_{k-2}^*}(\nu_{Z_{k-1}^*}(\mu_{H(S)}(s)))).$$

By Lemma 5, we know that $\nu_{Z_{k+1}^*}(\nu_{Z_k^*}(s_i)) = \nu_{Z_k^*}(\nu_{Z_{k+1}^*}(s_i))$. Hence,

$$s = \nu_{Z_{n-1}^*}(\nu_{Z_{n-2}^*}(\nu_{Z_{n-1}^*}(\mu_{H(S)}(s))))...$$

We can apply the same technique repeatedly until $Y_j$ becomes the first $\nu$. 

177
We now use Theorem 1 and Lemma 6 to develop our algorithm.

**Input:** a structure over one-level scheme $S = X_1^* \ldots Y_n^*$.

**Output:**
- true if $s$ is a one-level nested relation.
- false if $s$ is not a one-level nested relation.

**Algorithm:** 1NR

function 1NR $(s, S)$ : boolean;
begin
$W := H(S)$;
$V := X$;
OK_so_far := true;
while ($W \neq \emptyset$) and OK_so_far do
begin
find $Z^* \in W$ such that $X < H(S) - Z^* > \rightarrow Z^*$ holds in $s$;
if such $Z^*$ does not exist then
OK_so_far := false
else
$W := W - Z^*$;
$V := V \cup Z^*$;
end
end;
1NR := OK_so_far
end; {of 1NR}

Theorem 3: Let $s$ be a structure over a one-level scheme $S = X_1^* \ldots Y_n^*$. Algorithm 1NR correctly determines whether $s$ is a one-level nested relation.

**Proof:** The proof will be by induction on $n$.

**Basis:** $n = 1$. Then $S = X_1^*$. By Lemma 3b, $s = \nu_1(\alpha_1^*(s))$ if and only if $X \rightarrow Y_1^*$ holds in $s$. The algorithm 1NR clearly covers this case.

**Induction Step:** Assume Algorithm 1NR is correct for structures having $n-1$ higher order objects. From Theorem 1 we know that if $s$ is a 1NR then there must exist a $Z^* \in H(S)$ such that $s$ satisfies $X < H(S) - Z^* > \rightarrow Z^*$. If such a $Z^*$ does not exist we know that $s$ is not a nested relation over $S$. In this case Algorithm 1NR halts with result false. Otherwise, by Lemma 6 we know that if $s$ is a nested relation there exists a nesting sequence of $s$ starting with $Z$. We can therefore choose $Z$ as the first set to nest over. We now wish to treat $Z^*$ as an attribute rather than a higher order object. For simplicity, let $Z^* = Y_1^*$. Furthermore, let $X' = XZ^*$. In this setting $s$ is a structure over the one-level scheme $S' = XY_2^* \ldots Y_n^*$. As a consequence of the above reasoning, $s$ is a 1NR over $S$ if and only if $s$ is a 1NR over $S'$. Since $S'$ has only $n-1$ higher order objects, the induction hypothesis means that the Algorithm would give the correct answer if started on $(s, S')$. But when the Algorithm started on $(s, S)$ reaches the while statement the second time $V$ and $W$ contain the same values that the Algorithm started on $(s, S')$ would have the first time it reached the while statement. From this point the two cases run identically and give the same answer.

We claim the time complexity of Algorithm 1NR is $O(n^2p^2(m+q^2))$, where $m = |A(S)|$, $n = |H(S)|$, $p = |S|$, and $q = \max_t \in s, M \in H(S)(t(M))$. Notice that this is a crude upper bound.

6. Discussion and Future Research

The main contribution of this paper is a characterization of one-level nested relations in terms of a new family of dependencies, the strong functional dependencies. As a consequence of this characterization we were able to develop a polynomial-time algorithm to test whether a structure defined over a one-level scheme is a one-level nested relation. In this section we mention some open problems and directions for future research related to questions raised in this paper.

**Multi-level Nested Relations**

In this paper we only dealt with structures defined over one-level schemes. A natural question is: is it possible to extend our results to structures defined over an arbitrary multi-level scheme, i.e., can we find a simple characterization for multi-level nested relations?
Subclasses of Nested Relations

Roth, Korth and Silberschatz introduced a special class of nested relations, the so called PNF (partitioned normal form) relations [RKS]. A structure $s$ over scheme $S$ is a PNF relation if and only if $s$ satisfies the FD $A(S) \rightarrow S$ and for all $v \in s$ and all $M \in H(S)$, $v(M)$ is a PNF relation. It can be shown that a PNF relation is a nested relation but not conversely. The structure in Example 1 is a PNF relation, while $r_5$ in Figure 5 is a nested relation but not a PNF relation. PNF relations are closed under unnesting, a desirable property since in general nested relations are not closed under unnesting (cf. Example 3). For one-level nested relations, we have shown that the class in which the nesting is fully permutable is the largest subclass of INR which is closed under unnesting.

Attribute Addition

Consider the structure $r_8$ defined over scheme $POINT\star AREA$ of Example 4. We observed that $\nu_{\text{POINT}}(\mu_{\text{POINT}}(r_8)) \neq r_8$. The reason is that $r_8$ violates the FD $\text{AREA} \rightarrow \text{POINT}\star$. Suppose however that for some reason we still would like to normalize $r_8$. The only reasonable thing to do, if we want to avoid information loss, is to introduce an extra attribute $I$ and extend $r_8$ to a structure $r_8^*$ over scheme $I, \text{AREA}, \text{POINT}\star$ such that $r_8^*$ satisfies the FD $I \rightarrow \text{AREA} \rightarrow \text{POINT}\star$ and $r_8$ can be obtained as a projection of $r_8^*$. A possible extension of $r_8$ is shown in Figure 11. Clearly such an extension is not unique.

<table>
<thead>
<tr>
<th>$I$</th>
<th>$\text{POINT}\star$</th>
<th>$\text{AREA}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(0,0)$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$(1,0)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$(0,2)$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$(0,0)$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$(2,0)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$(0,1)$</td>
<td></td>
</tr>
</tbody>
</table>

Figure 11

This example illustrates the coupling between attribute addition and horizontal decomposition. It would be interesting to know the role of NEST and UNNEST operators with regard to this problem.

Normalization and SFDs

The following example is from Ullman ([Ull] Example 5.10). Consider the scheme $R = \text{CITY,ST,ZIP}$. A tuple $(c,s,t)$ is in a relation over $R$ if city $c$ has a building with street address $s$, and $z$ is the zip code for that address in the city. It is assumed that the nontrivial dependencies are:

$$\text{ZIP} \rightarrow \text{CITY}$$

$$\text{CITY } \text{ST} \rightarrow \text{ZIP}$$

It is well known that there exists no dependency-preserving decomposition of $R$ into schemes which are in BCNF with respect to the above set of dependencies. Consider the relation $r_{12}$ shown in Figure 12; $r_{12}$ satisfies the given FDs.

<table>
<thead>
<tr>
<th>CITY</th>
<th>ST</th>
<th>ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$s_1$</td>
<td>$z_1$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$s_2$</td>
<td>$z_1$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$s_3$</td>
<td>$z_1$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$s_4$</td>
<td>$z_2$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$s_5$</td>
<td>$z_2$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$s_1$</td>
<td>$z_2$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$s_4$</td>
<td>$z_3$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$s_6$</td>
<td>$z_3$</td>
</tr>
</tbody>
</table>

Figure 12

Consider the structure $r_{13} = \nu_{\text{ST}}(r)$ over the scheme $\text{CITY,ST}\star, \text{ZIP}$ shown in Figure 13; $r_{13}$ satisfies the SFDs

$$\text{ZIP} \rightarrow \text{CITY } \text{ST}\star$$

$$\text{CITY } <\text{ST}\star> \rightarrow \text{ZIP}$$

179
Figure 13

Nesting on ST preserves the dependencies (if we are willing to translate FDs into SFDs in the nested relation). Furthermore, the update anomalies apparent in $r_{12}$ disappear in $r_{13}$. There appear to be many other cases where nested relations are a good alternative to normal forms over flat relations (cf. [FV3]). Characterizing the situations where nesting produces good database design could be very worthwhile.

Bibliography


